
edureka!
Hadoop Tutorial

Agenda

edureka!

- Big Data Growth Drivers
- What is Big Data?
- Hadoop Introduction
- Hadoop Master/Slave Architecture
- Hadoop Core Components
- HDFS Data Blocks
- HDFS Read/Write Mechanism
- What is MapReduce
- MapReduce Program
- MapReduce Job Workflow
- Hadoop Ecosystem
- Hadoop Use Case: Analyzing Olympic Dataset

Big Data Growth Drivers

Big Data Growth Drivers

edureka!

Data Generated
Every 60 Seconds

Users like
4,166,667
posts

Users send
347,222
tweets

Users cast
18,327
votes

Users like
1,736,111
posts

Users upload
300 hours
of new video

Global Mobile Data Traffic, 2015 to 2020

edureka!

Cisco Forecasts 30.6 Exabytes per Month of Mobile Data Traffic by 2020

3 major trends contributing to the growth of mobile data traffic:

- Adapting to Smarter Mobile Devices
- Defining Cell Network Advances—2G, 3G, and 4G (5G Perspectives)
- Reviewing Tiered Pricing—Unlimited Data and Shared Plans

Source: http://www.cisco.com/c/en/us/solutions/collateral/service-provider/visual-networking-index-vni/mobile-white-paper-c11-520862.html

What is Big Data?

What is Big Data?

edureka!

"Big data is the term for a collection of data sets so large and complex that it becomes difficult to process using on-hand database management tools or traditional data processing applications"

Volume

Processing increasing huge data sets

Variety

Processing different types of data

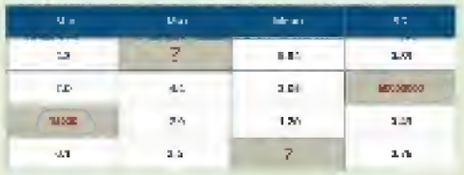
Velocity

Data is being generated at an alarming rate

Value

Finding correct meaning out of the data

Veracity



Uncertainty and inconsistencies in the data

Let us understand Problems with Big Data and Traditional System with a Story

Story of Big Data & Traditional System

edureka!

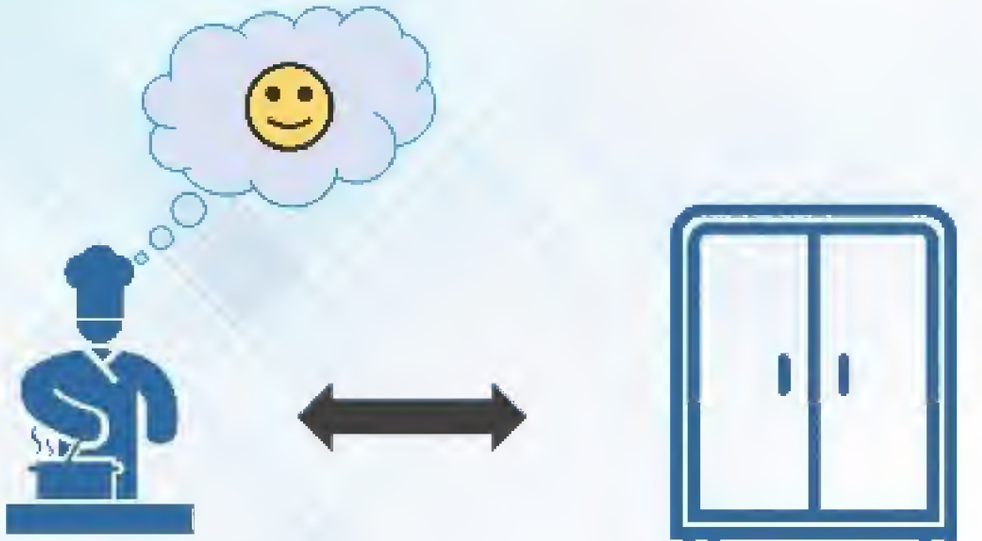
Scenario:
Bob has opened a small restaurant in his city

Traditional Scenario

edureka!

Traditional Scenario:

2 orders per hour

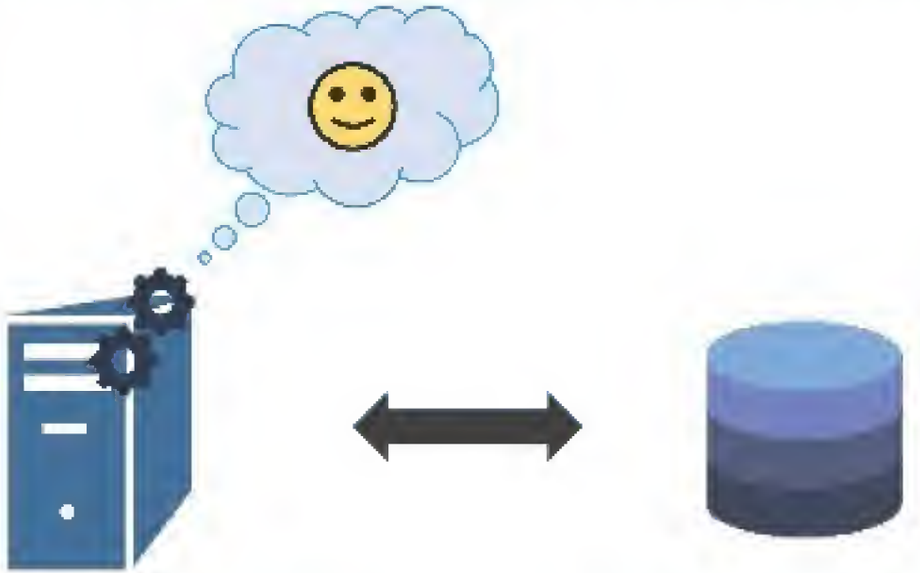


Single Cook

Food Shelf

Traditional Scenario:

Data is generated at a steady rate and is structured in nature



Traditional Processing System

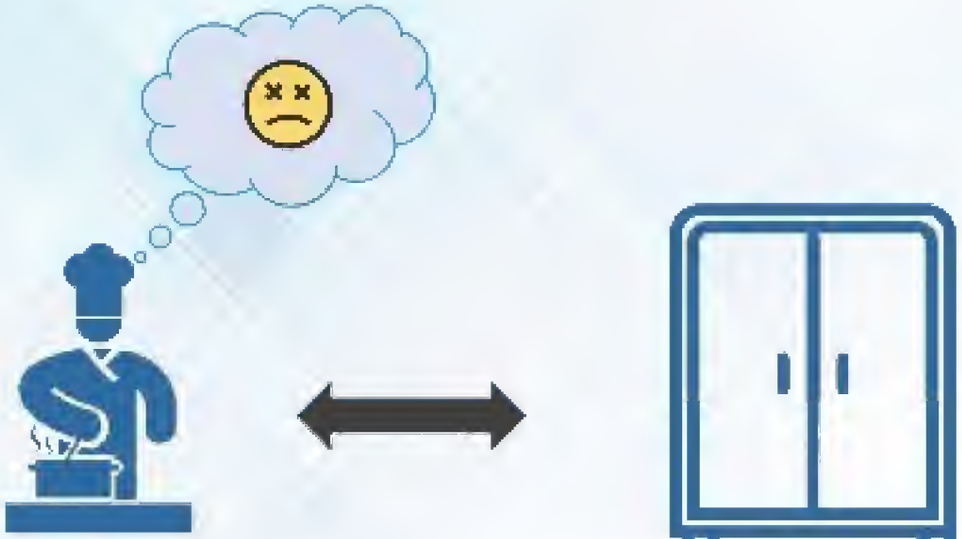
RDBMS

Failure of Traditional System

edureka!

Scenario 2:

- They started taking Online orders
- 10 orders per hour

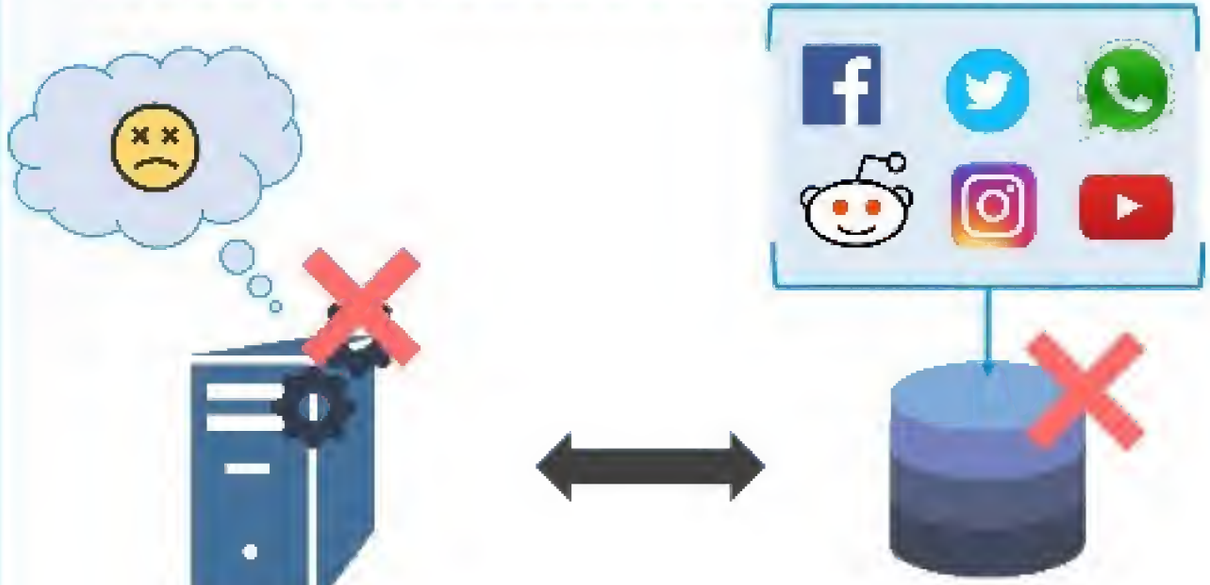


Single Cook
(Regular Computing System)

Food Shelf
(Data)

Big Data Scenario:

Heterogenous data is being generated at an alarming rate by multiple sources



Traditional Processing System

RDBMS

Issue 1: Too Many Orders Per Hour

Solution: Hiring Multiple Cook

Need of an Effective Solution

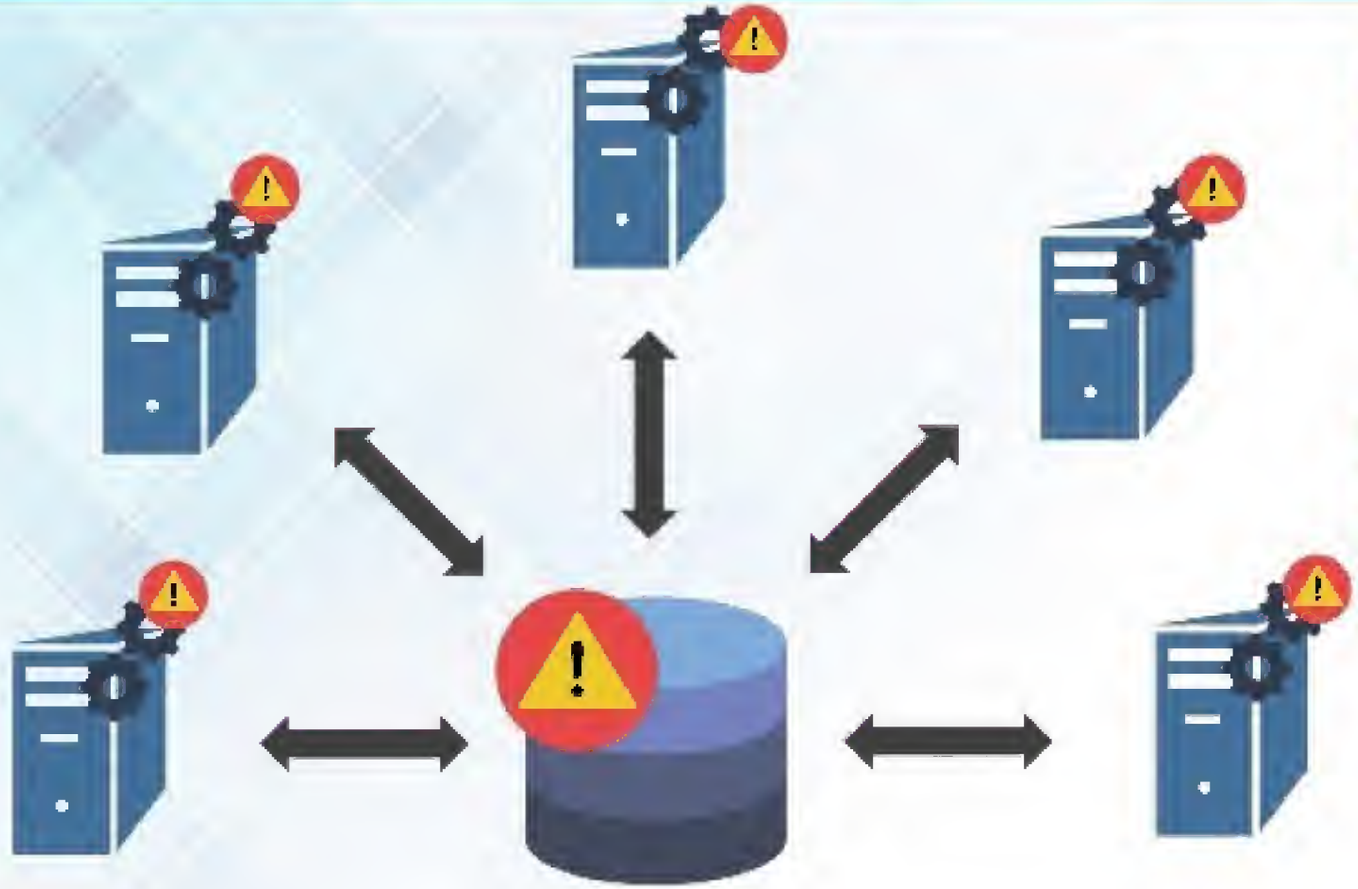
Scenario:

Multiple Cook cooking food

Issue:

Food Shelf becomes the BOTTLENECK

Need of an Effective Solution



Data Warehouse

Scenario:

Multiple Processing Unit for data processing

Issue:

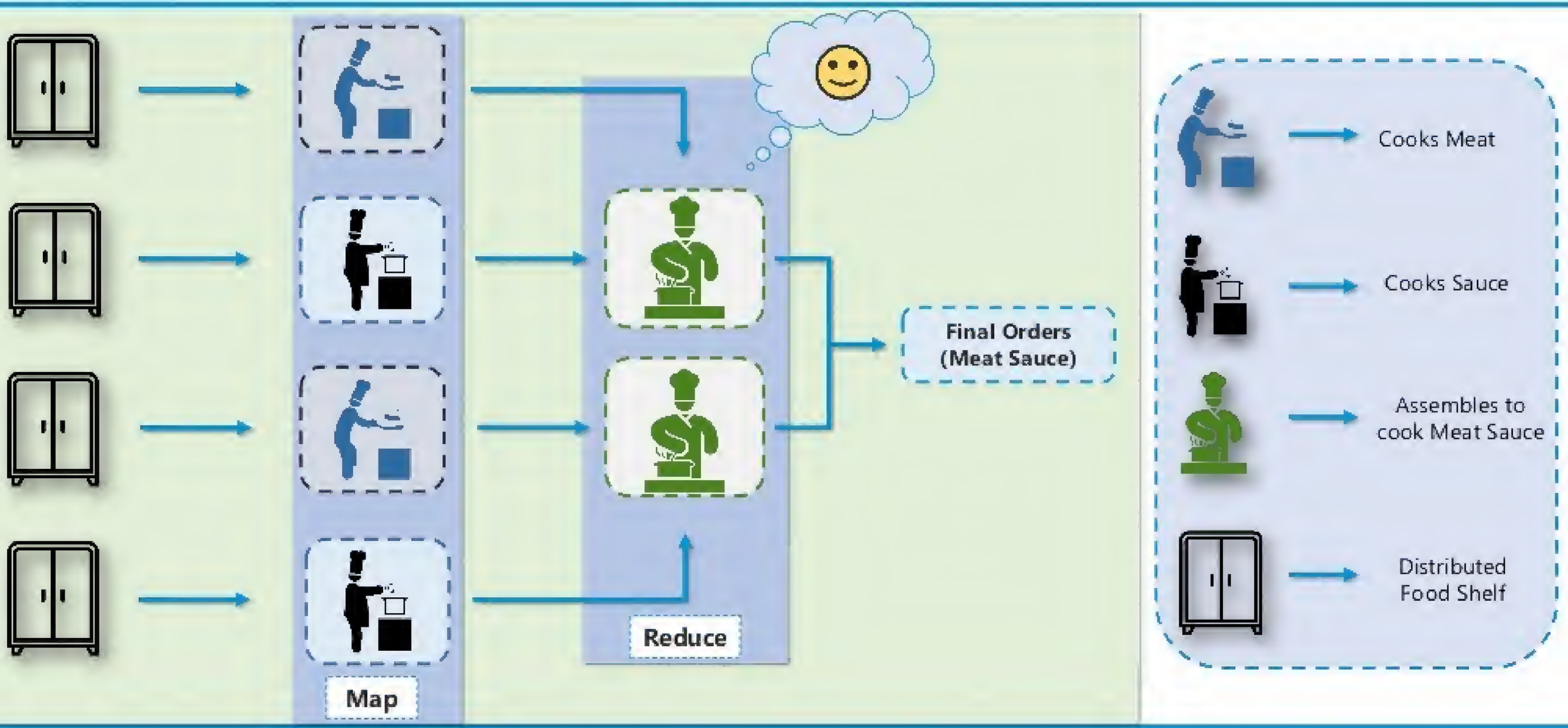
Bringing data to processing generated lots of Network overhead

Issue 2: Food Shelf becomes the Bottleneck

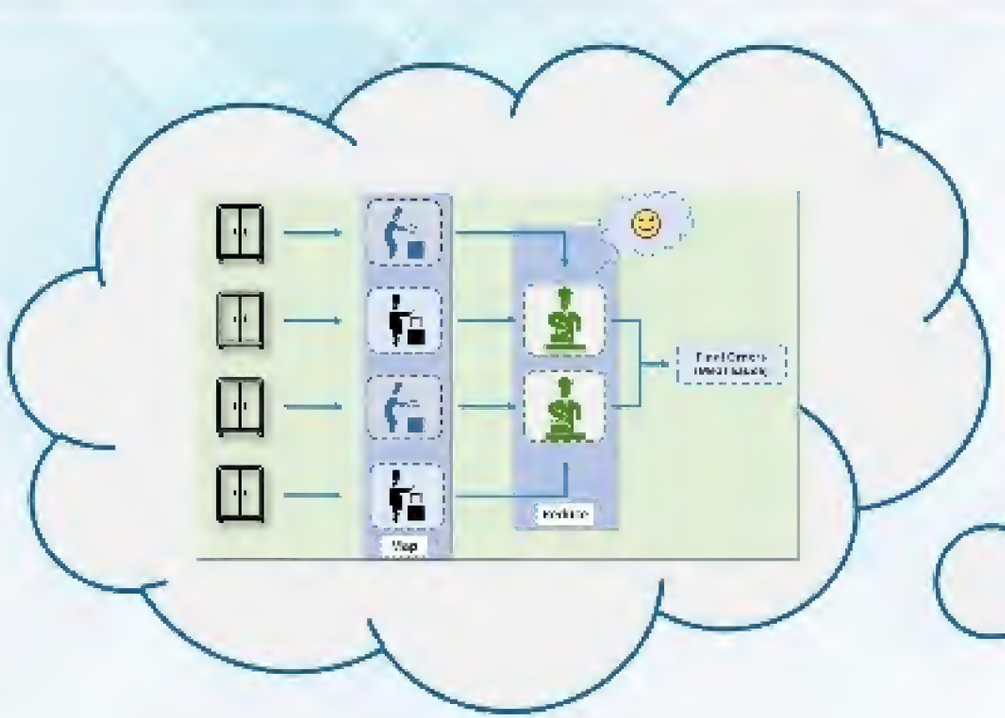
Solution: Distributed and Parallel Approach

Effective Solution

edureka!



Need of a Framework



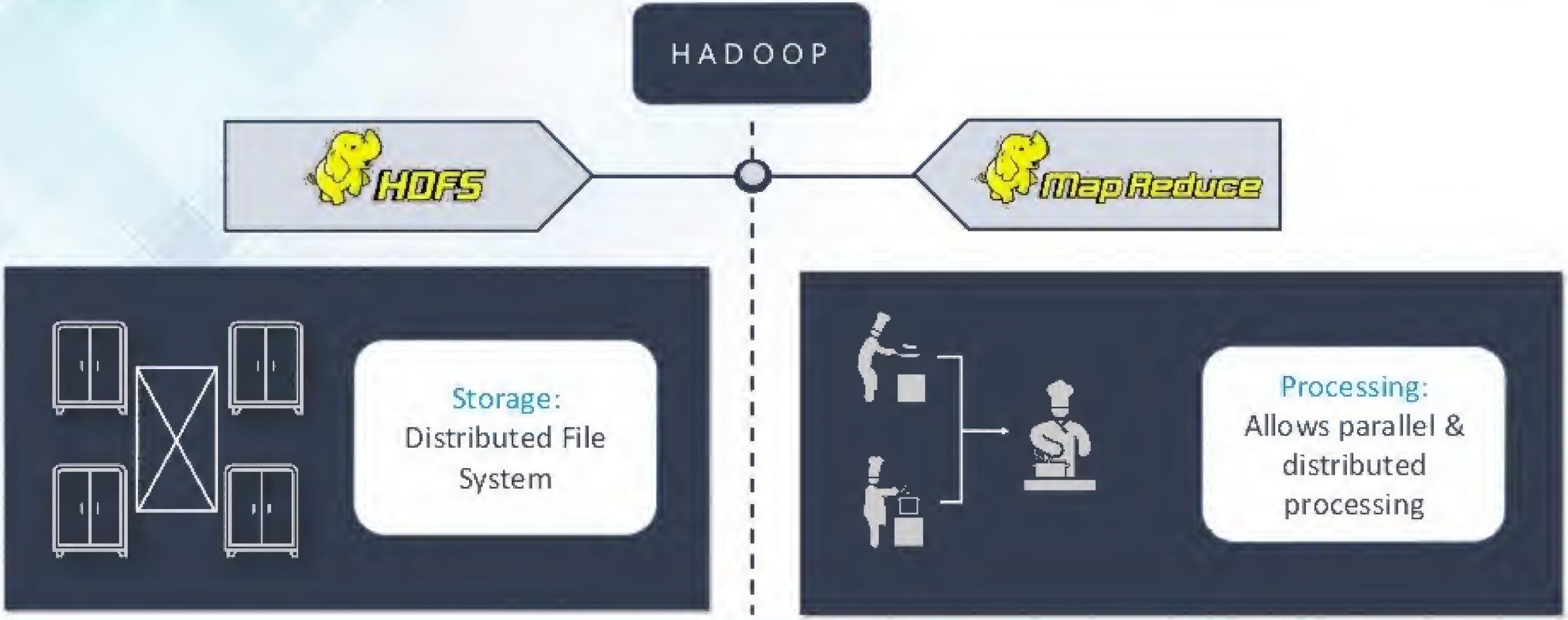
Do we have a framework that works like that ?

Apache Hadoop:
Framework to Process Big Data

Apache Hadoop: Framework to Process Big Data

edureka!

Hadoop is a framework that allows us to store and process large data sets in parallel and distributed fashion



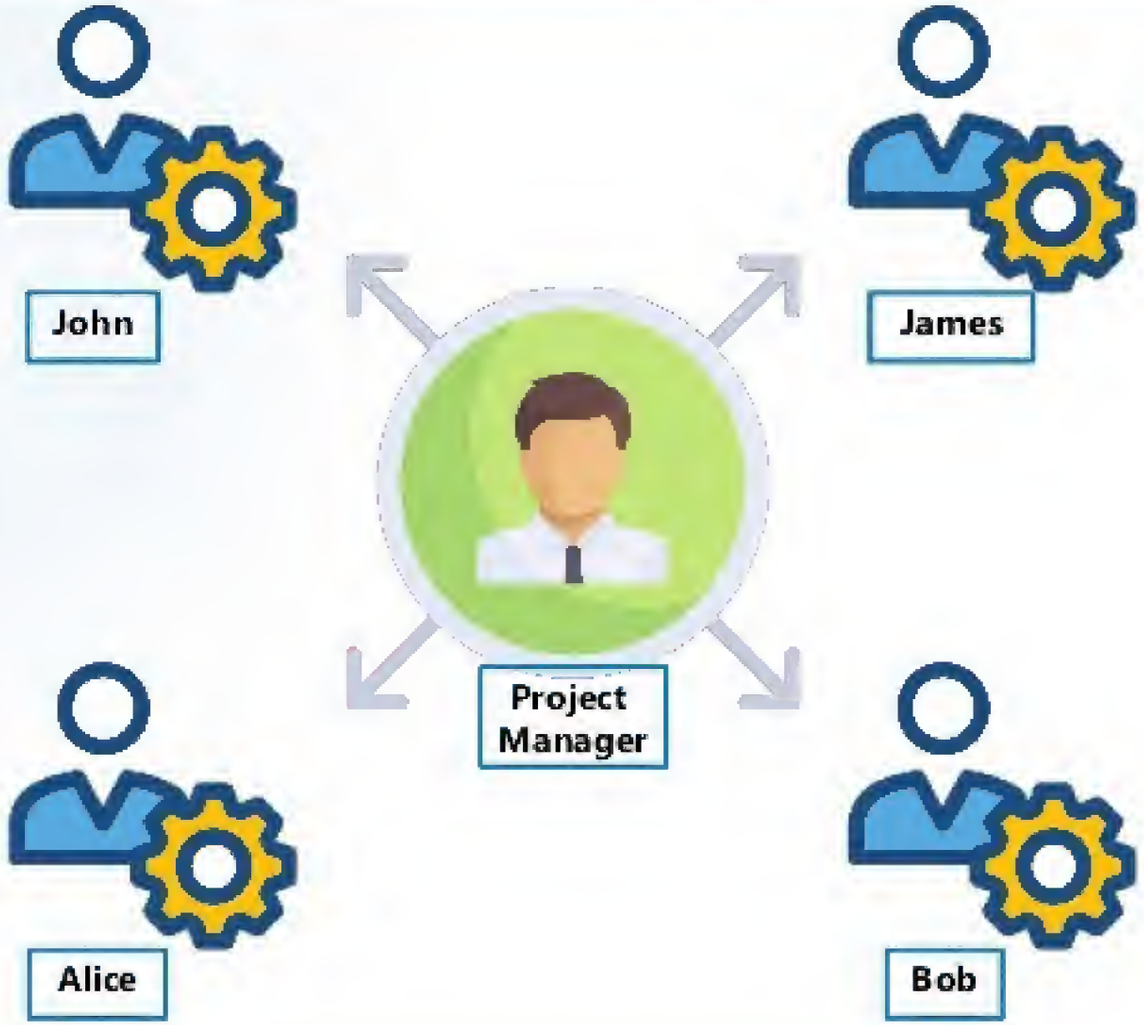
Hadoop: Master/Slave Architecture

Hadoop: Master/Slave Architecture

edureka!

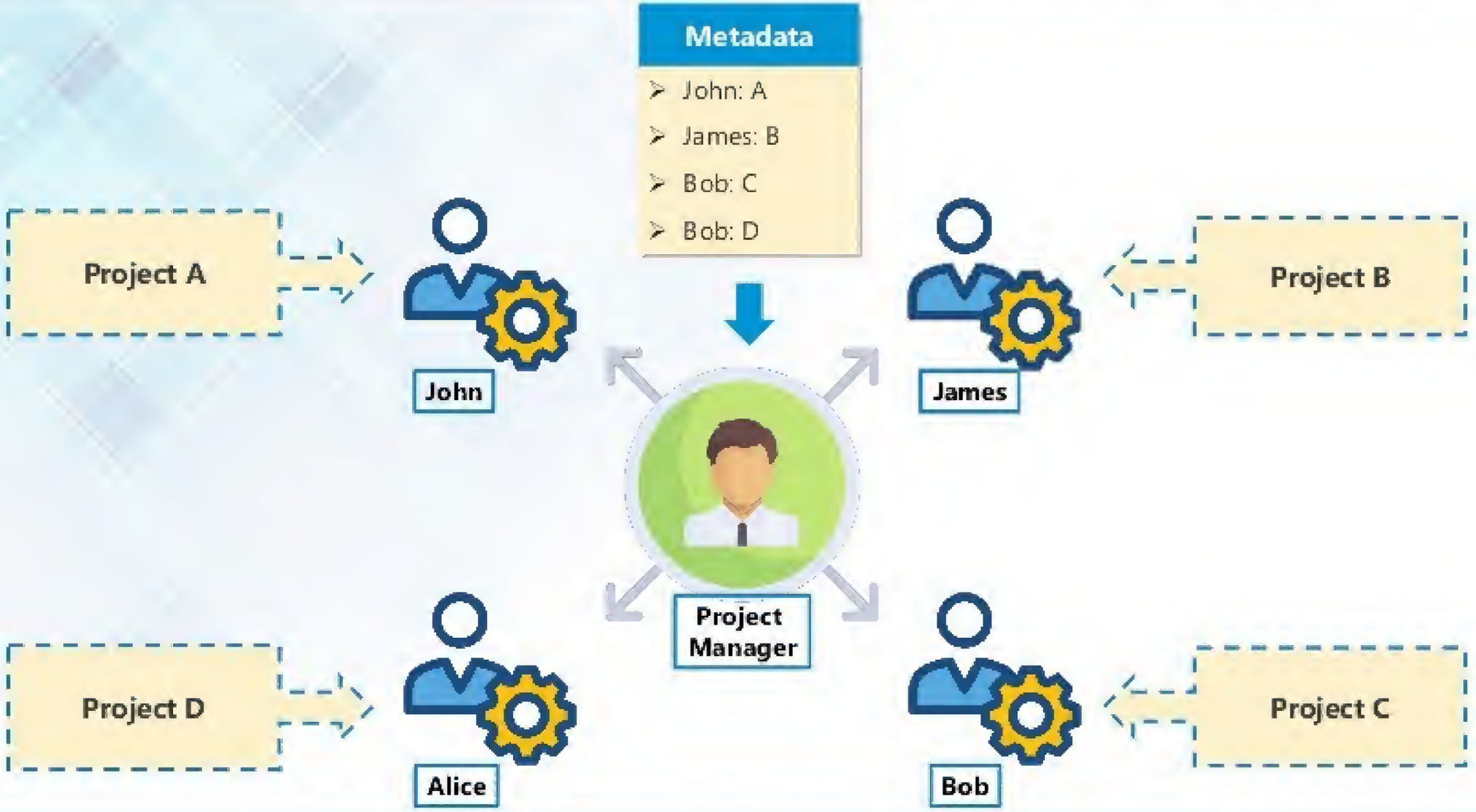
Scenario:

A project Manager managing a team of four employees. He assigns project to each of them and tracks the progress

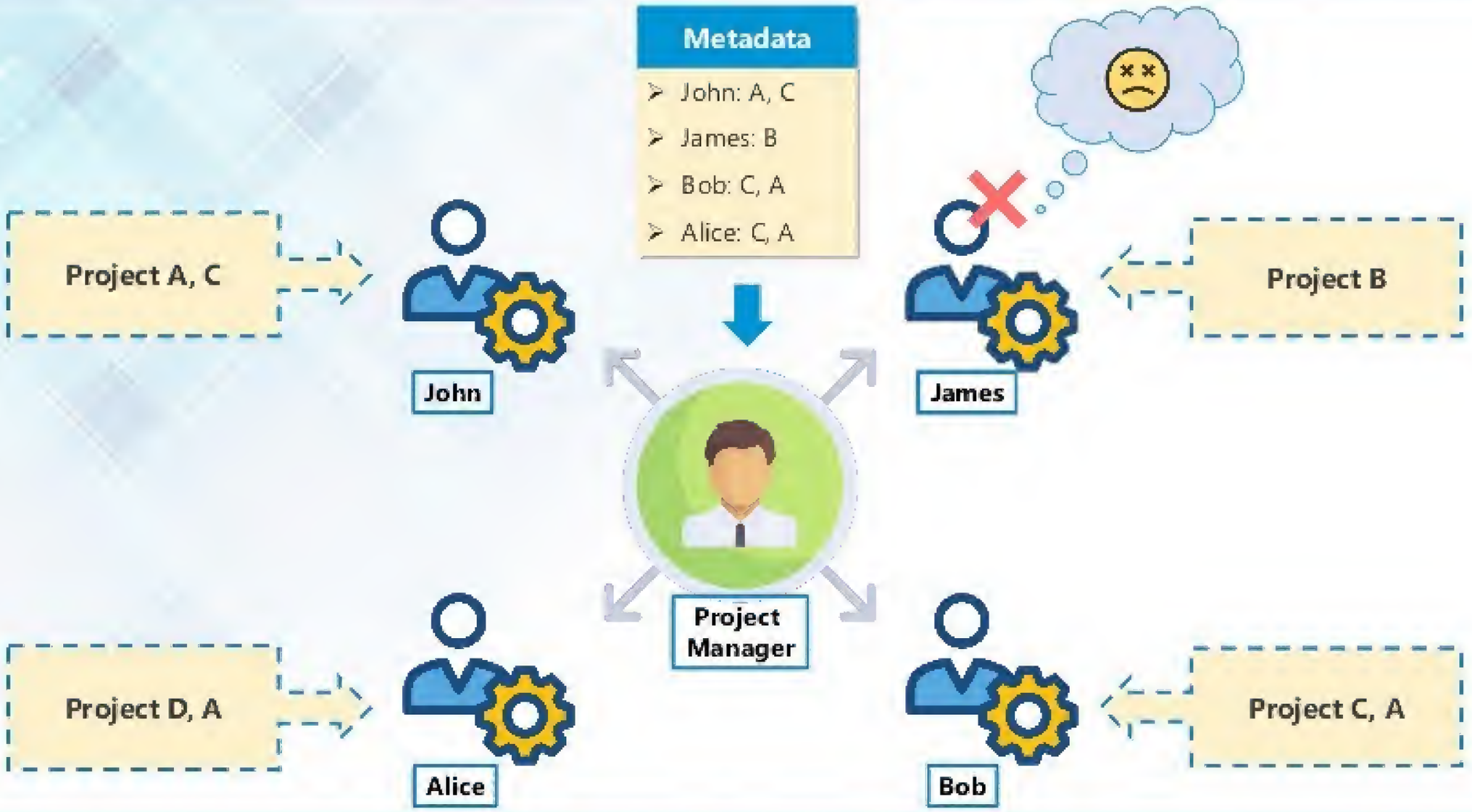


Hadoop: Master/Slave Architecture

edureka!

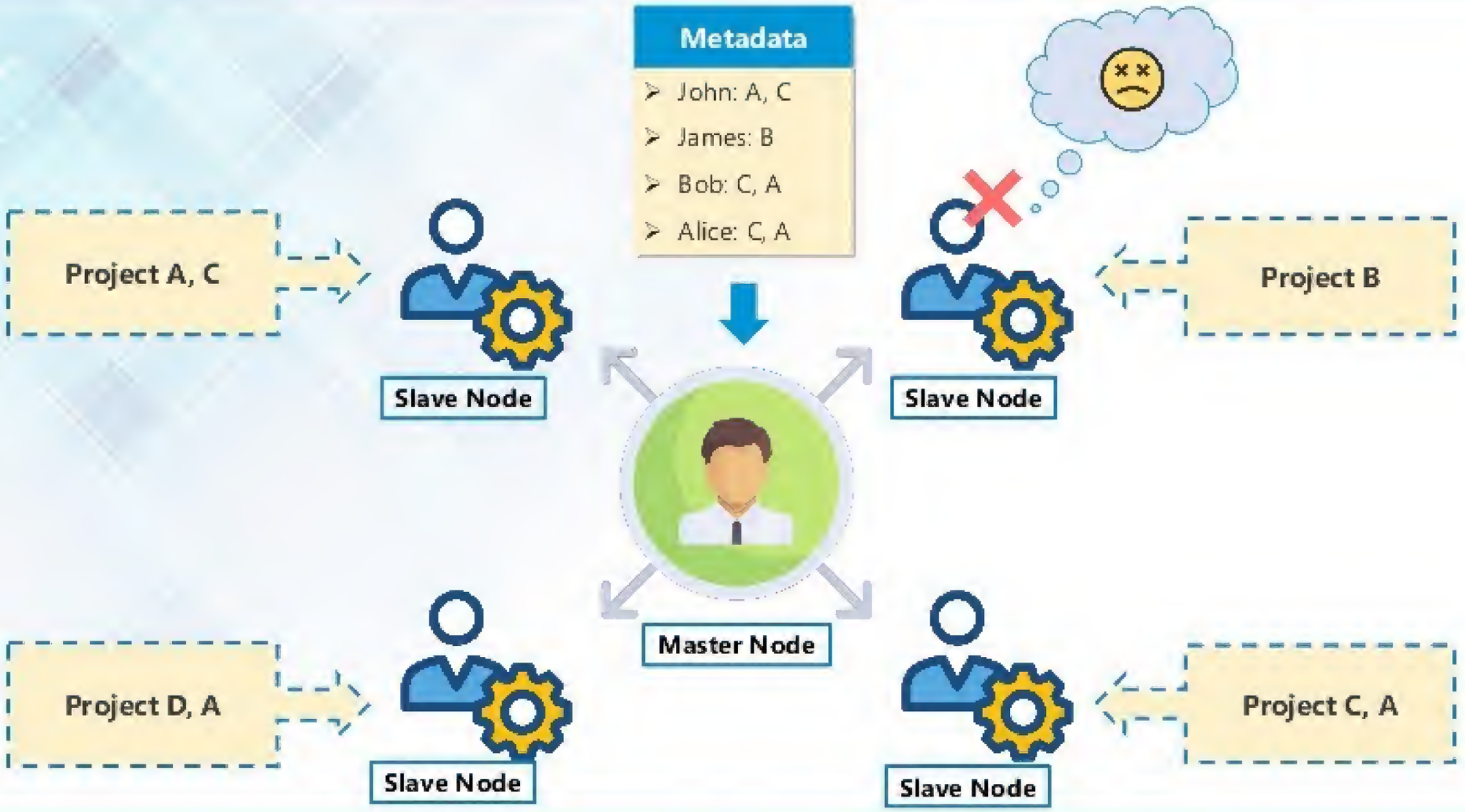


Hadoop: Master/Slave Architecture



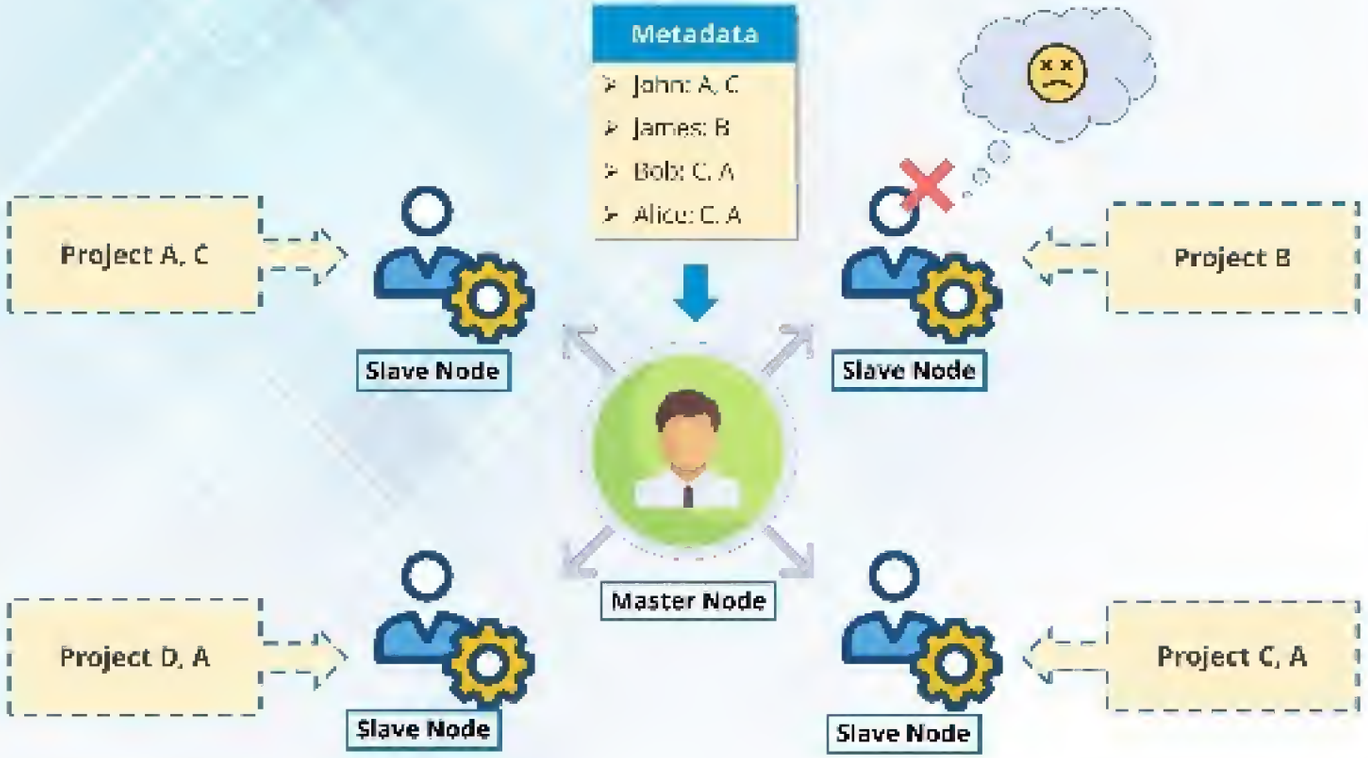
Hadoop: Master/Slave Architecture

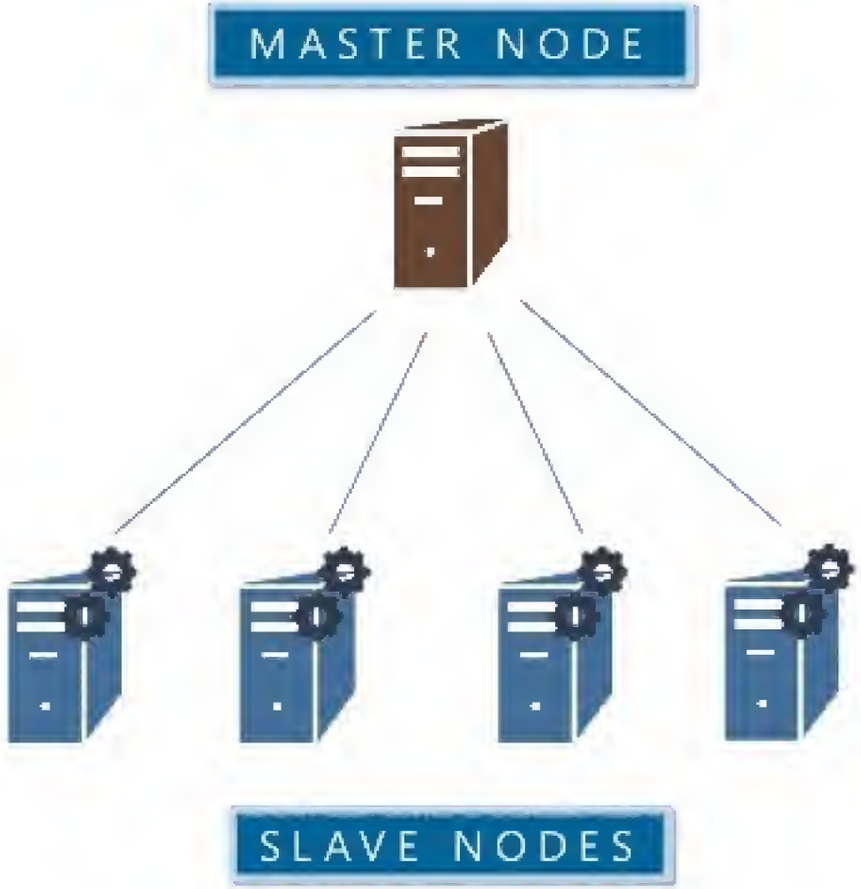
edureka!



Hadoop: Master/Slave Architecture

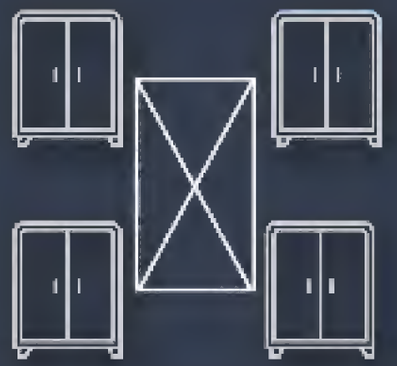
edureka!





HADOOP CORE COMPONENTS

HDFS



Storage:
Distributed File System

Allows parallel & distributed processing

HDFS Core Components:

01

NameNode

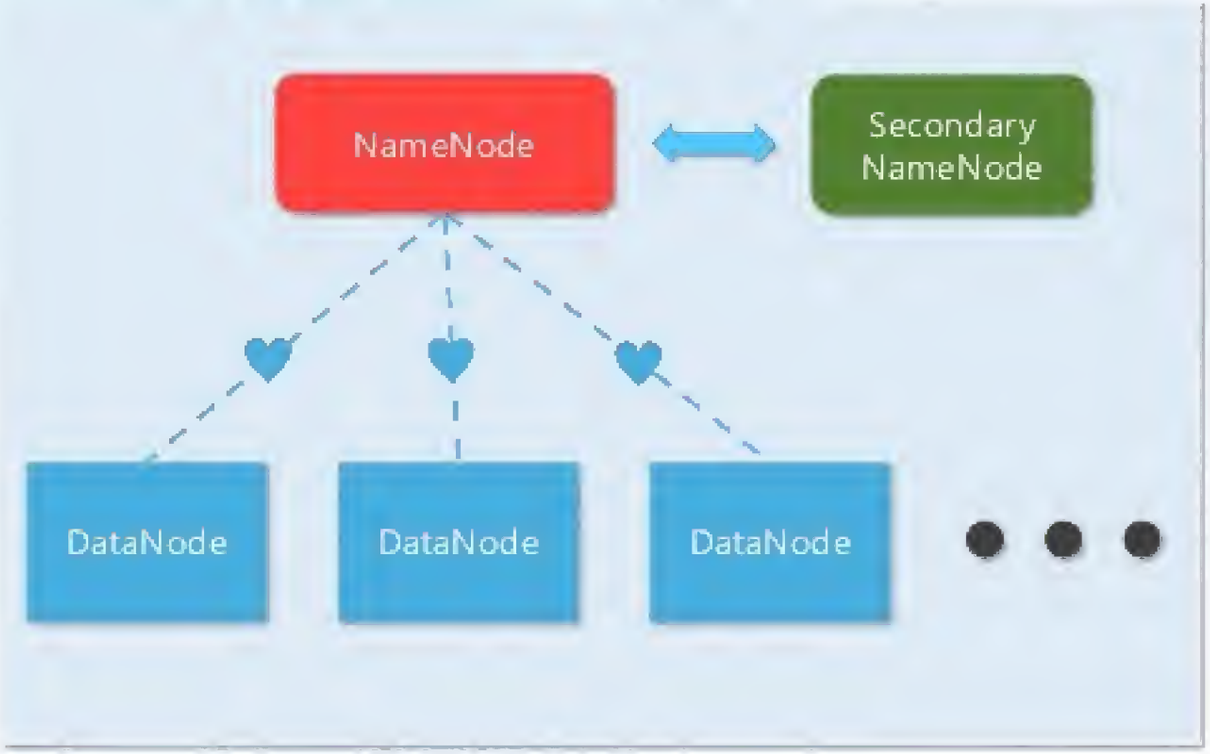
02

DataNode

03

Secondary NameNode

NameNode & DataNode



NameNode:

- Maintains and Manages DataNodes
- Records metadata i.e. information about data blocks e.g. location of blocks stored, the size of the files, permissions, hierarchy, etc.
- Receives heartbeat and block report from all the DataNodes

DataNode:

- Slave daemons
- Stores actual data
- Serves read and write requests from the clients

HDFS Core Components:

01

NameNode

02

DataNode

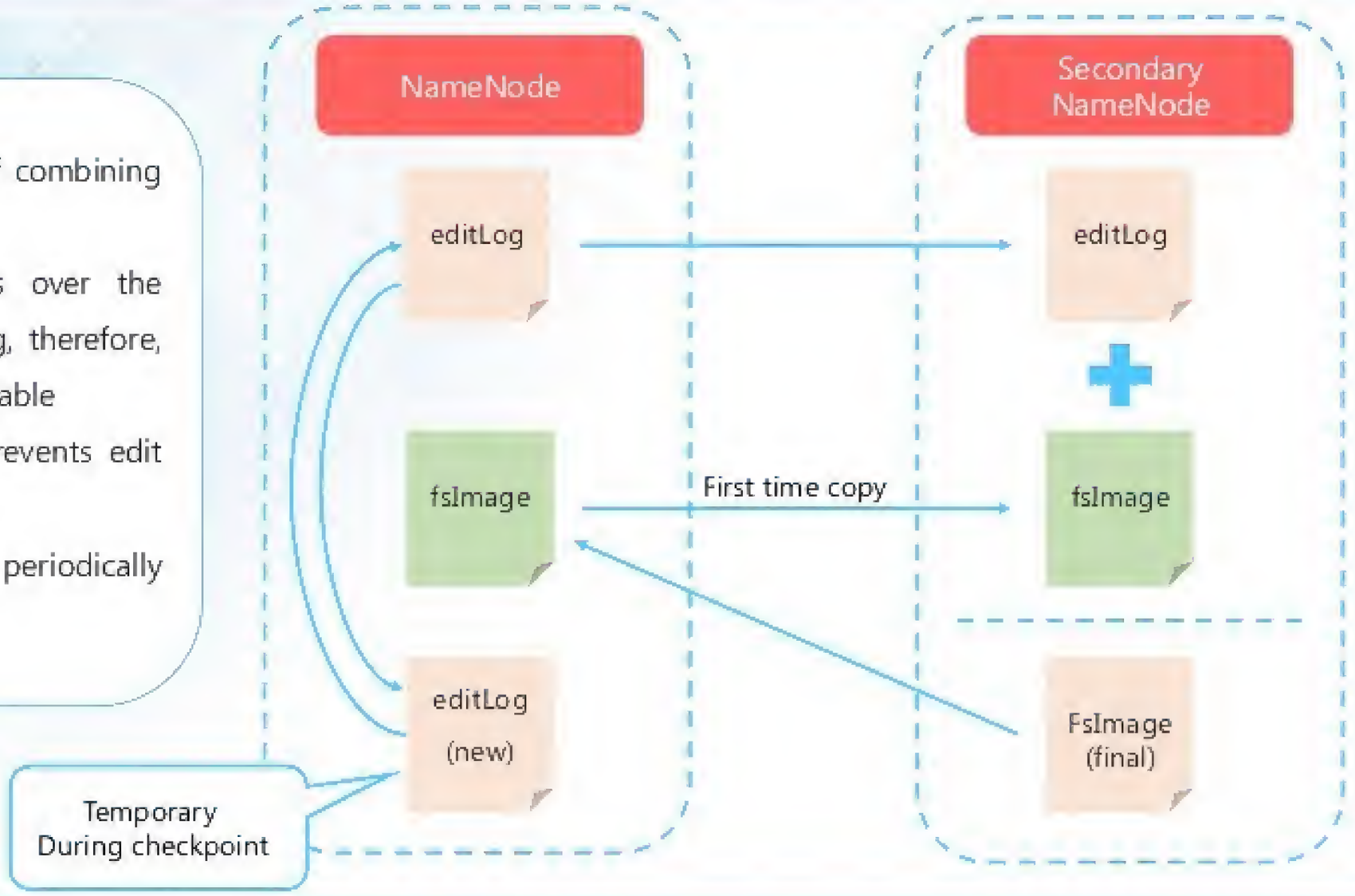
03

Secondary NameNode

Secondary NameNode & Checkpointing

edureka!

- Checkpointing is a process of combining edit logs with FsImage
- Secondary NameNode takes over the responsibility of checkpointing, therefore, making NameNode more available
- Allows faster Failover as it prevents edit logs from getting too huge
- Checkpointing happens periodically (default: 1 hour)

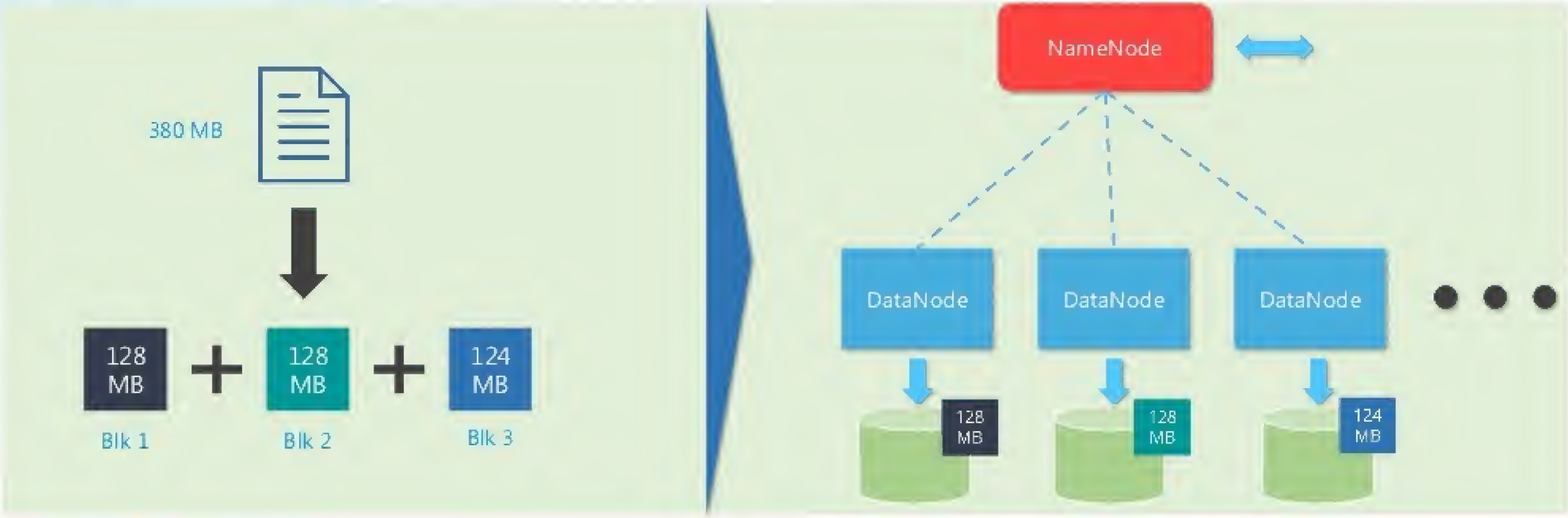


How the data is actually stored in DataNodes?
HDFS Data Blocks

HDFS Data Blocks

edureka!

- Each file is stored on HDFS as blocks
- The default size of each block is 128 MB in Apache Hadoop 2.x (64 MB in Apache Hadoop 1.x)



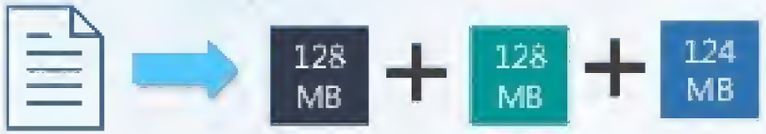
Fault Tolerance: How Hadoop cope up with DataNode Failure?

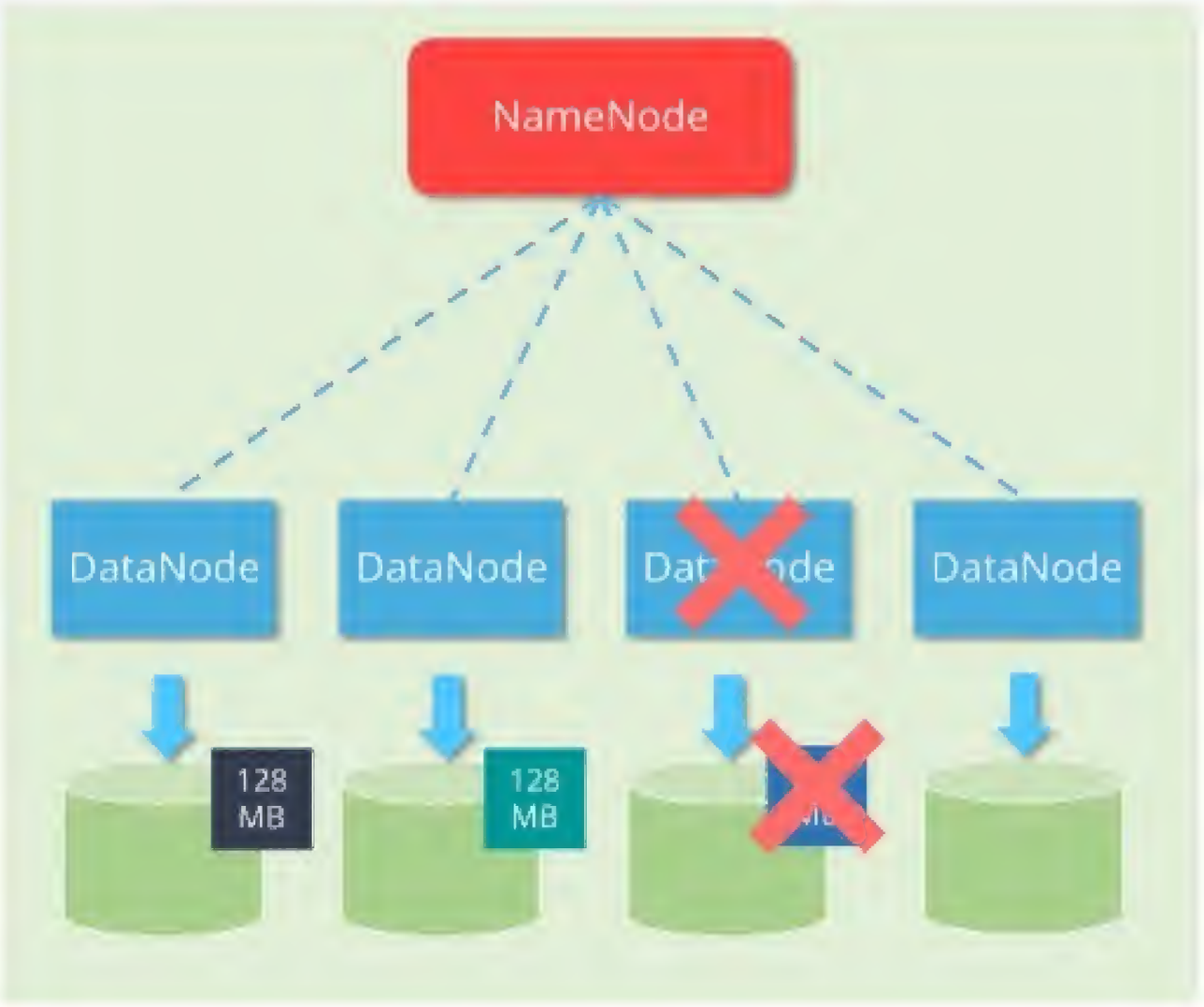
Fault Tolerance

edureka!

Scenario:

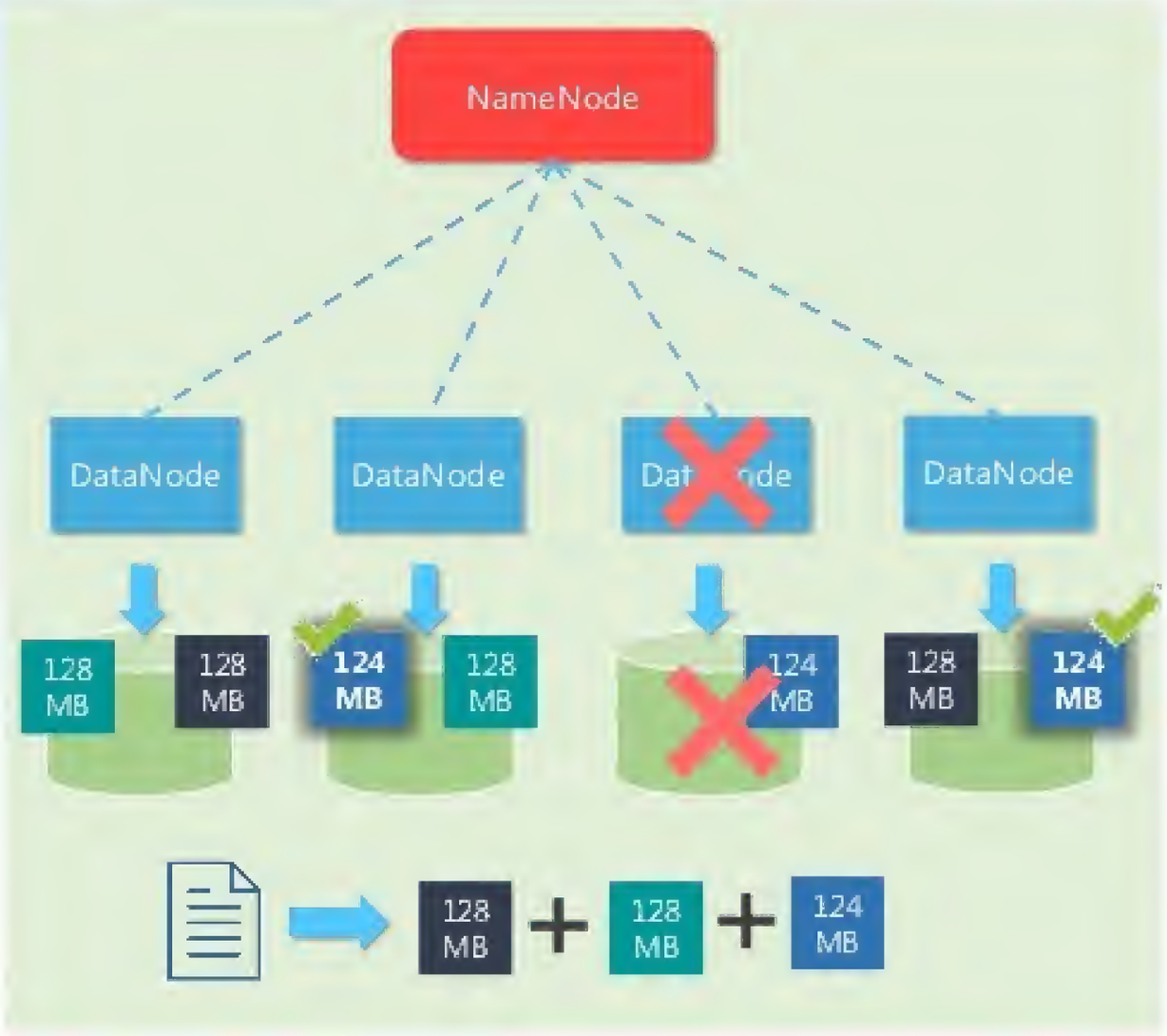
One of the DataNodes crashed containing the data blocks





Solution: Replication Factor

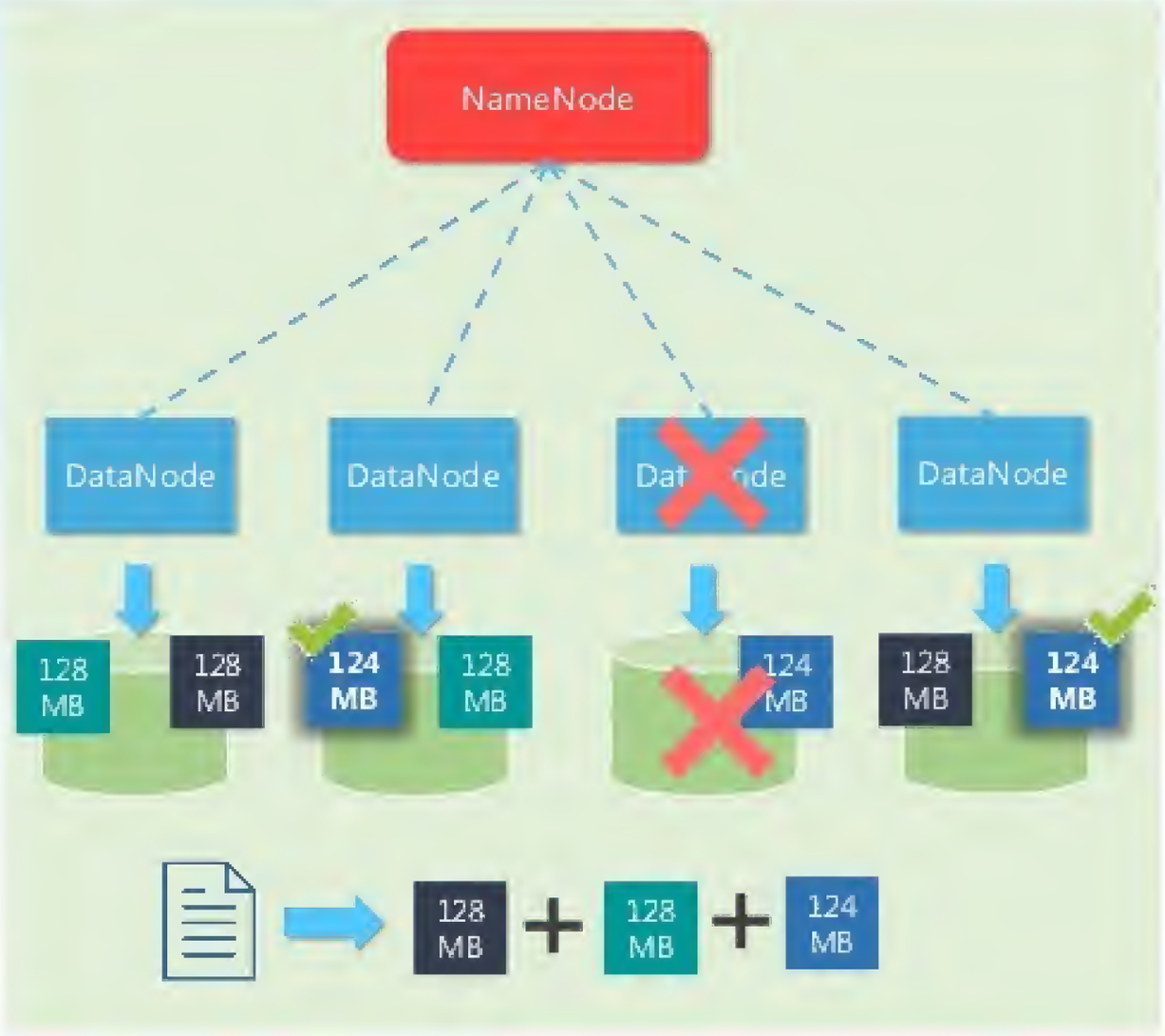
Fault Tolerance: Replication Factor



Solution:

Each data blocks are replicated (thrice by default) and are distributed across different DataNodes

Fault Tolerance: Replication Factor



Solution:

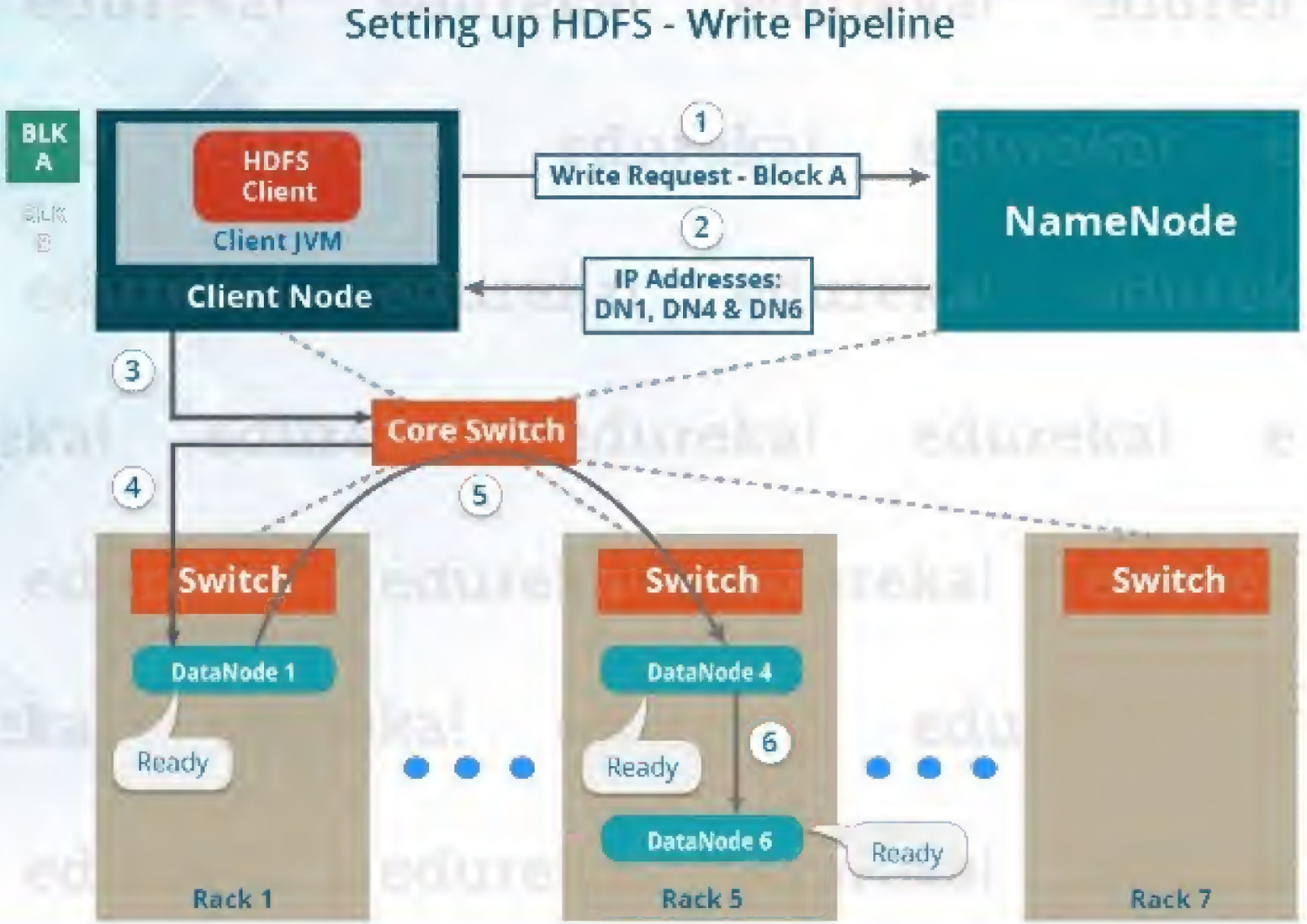
Each data blocks are replicated (thrice by default) and are distributed across different DataNodes

As it is said Never Put All Your Eggs in the Same Basket

HDFS Write Mechanism

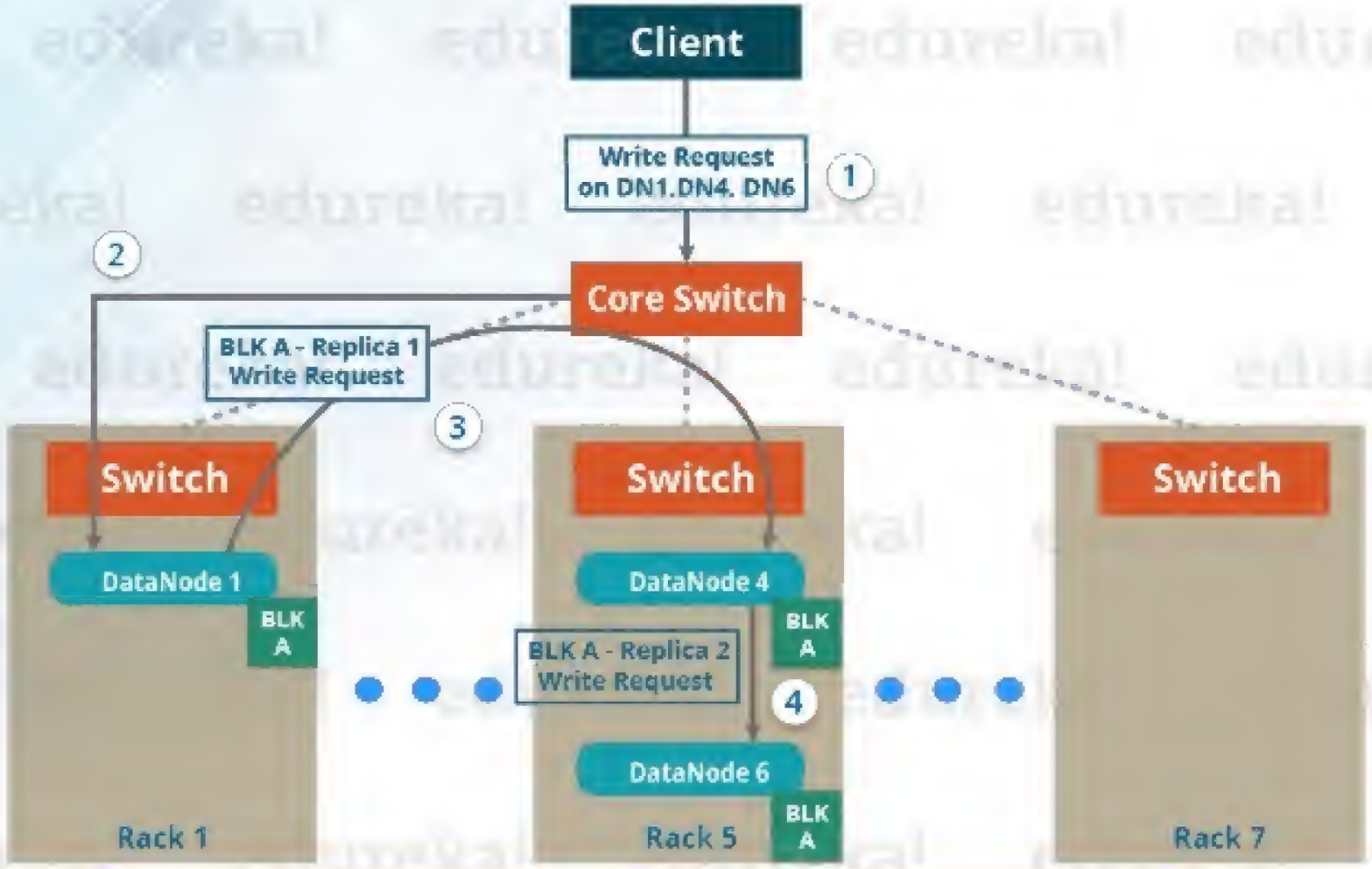
HDFS Write Mechanism – Pipeline Setup

edureka!



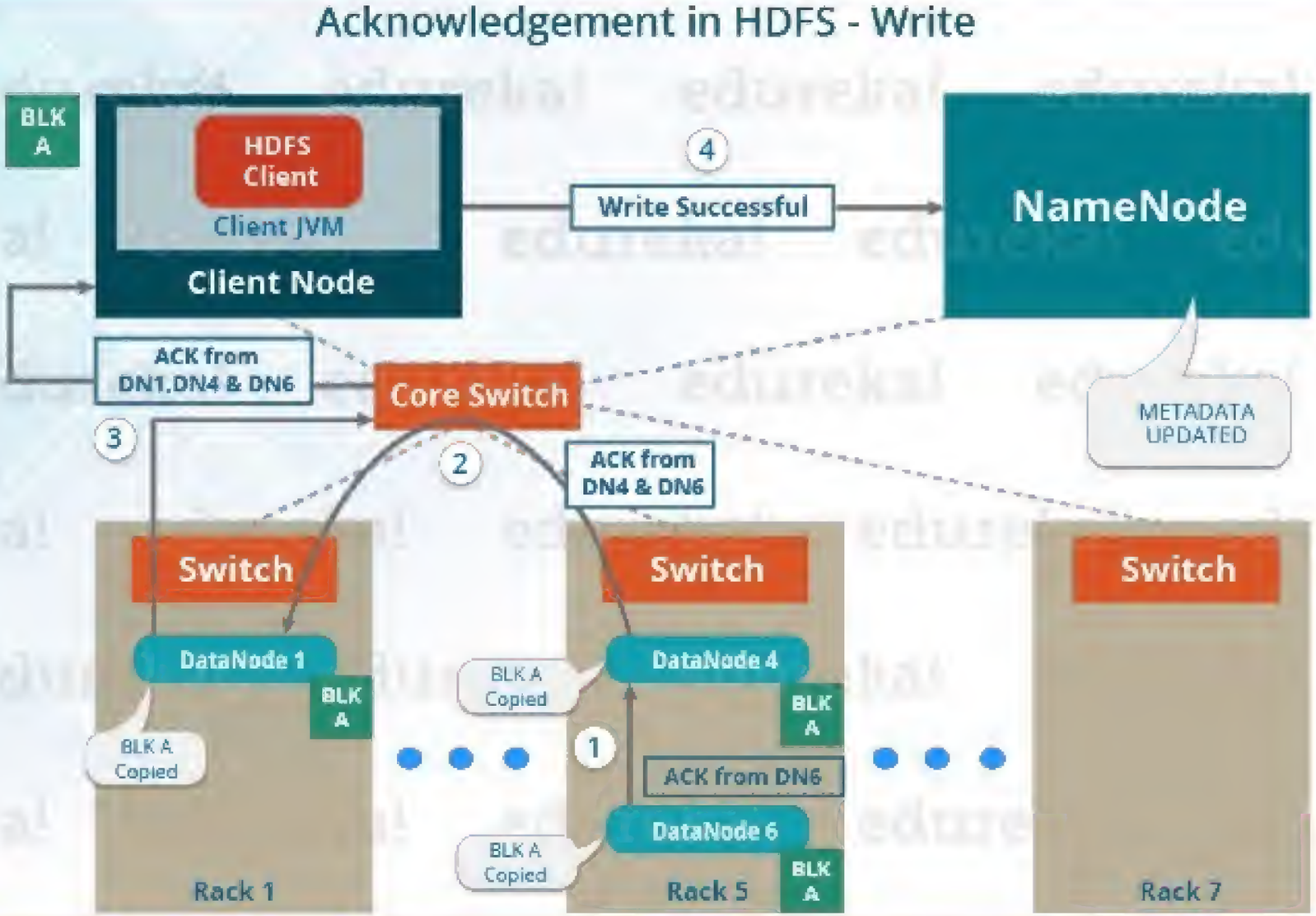
HDFS Write Mechanism – Writing a Block

edureka!



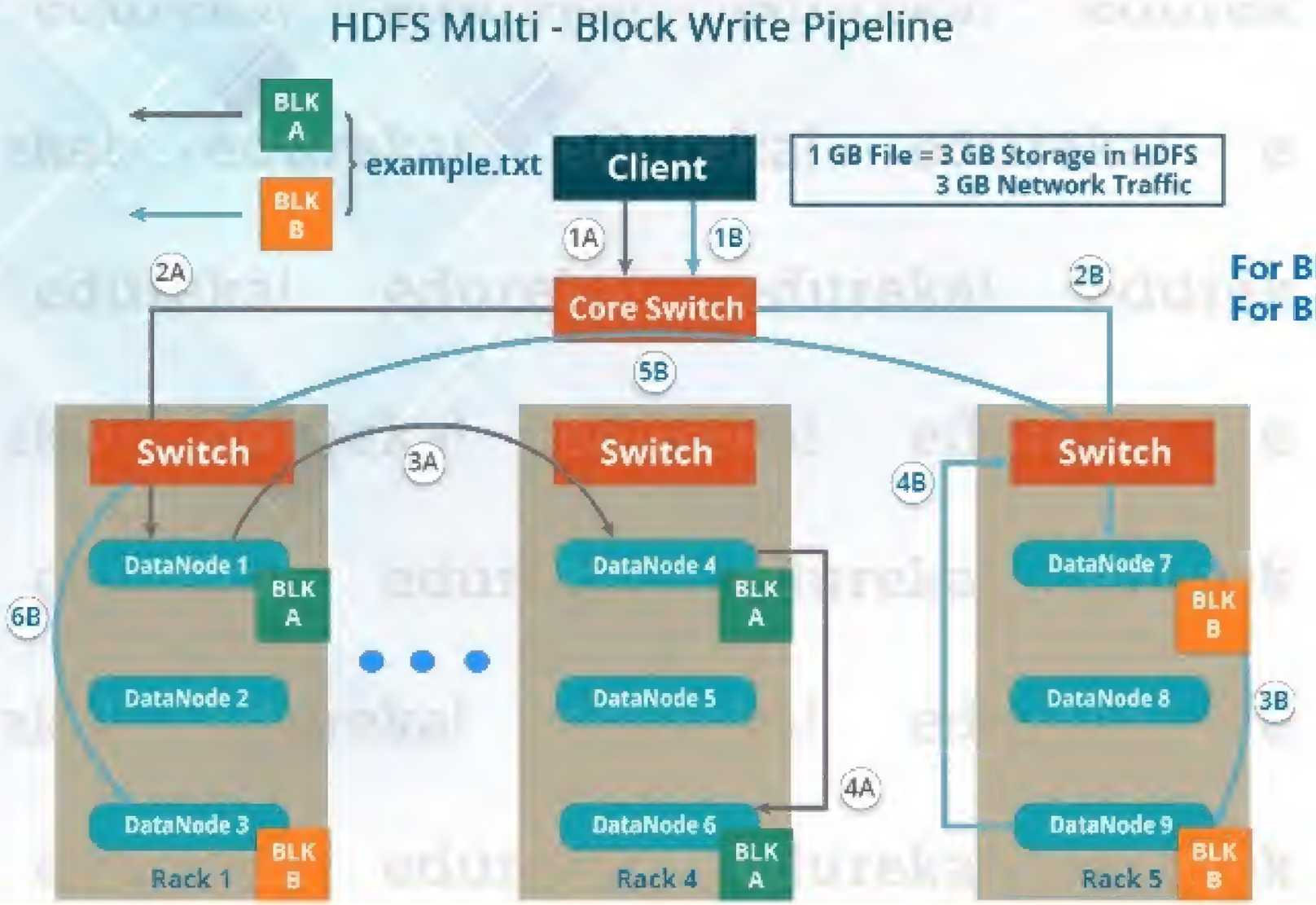
HDFS Write Mechanism - Acknowledgement

edureka!



HDFS Multi-Block Write Mechanism

edureka!

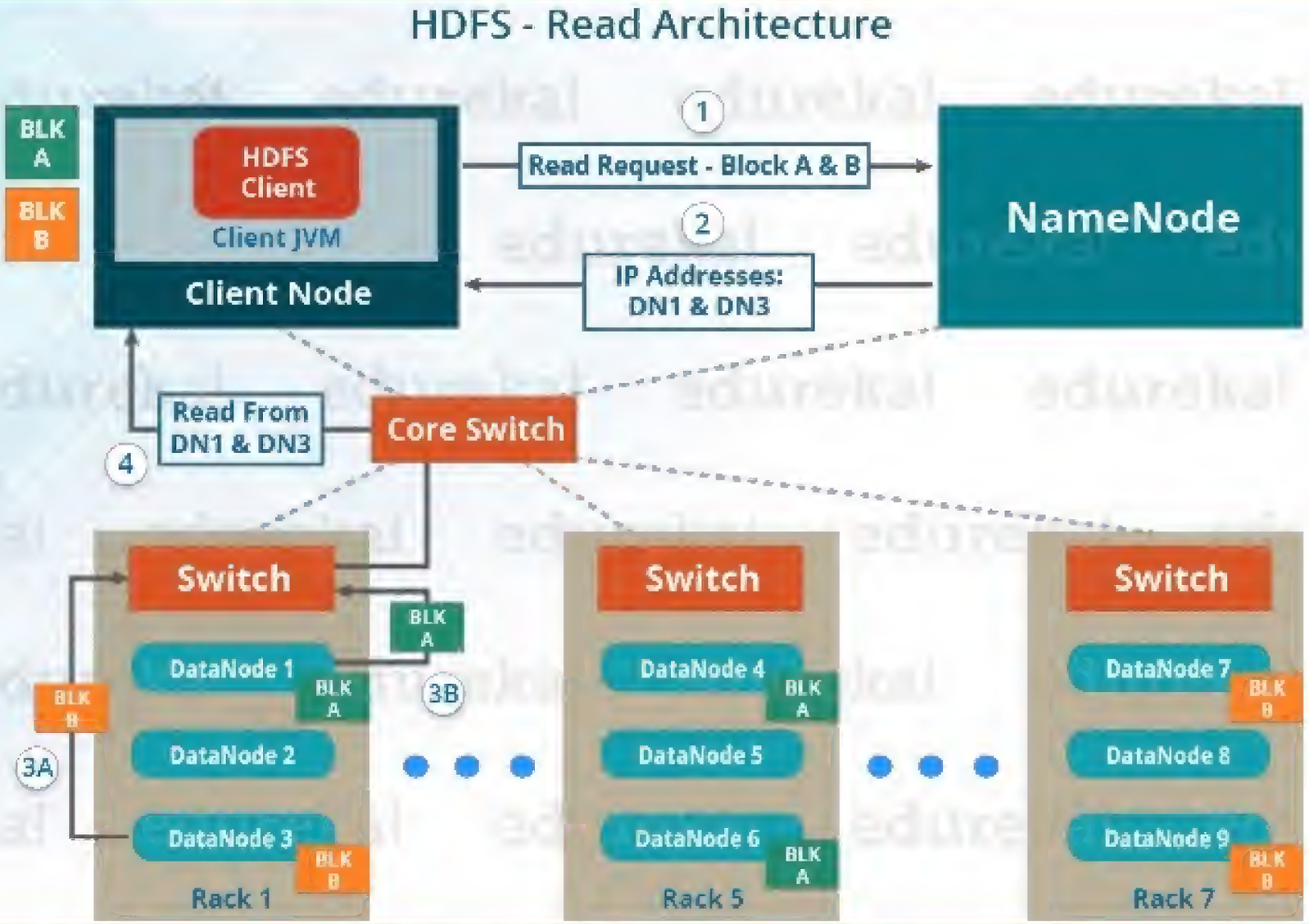


For Block A: 1A -> 2A -> 3A -> 4A
For Block B: 1B -> 2B -> 3B -> 4B -> 5B -> 6B

HDFS Read Mechanism

HDFS Read Mechanism

edureka!

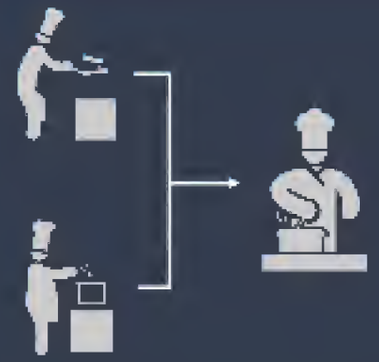


HADOOP CORE COMPONENTS

HDFS

Distributed File System

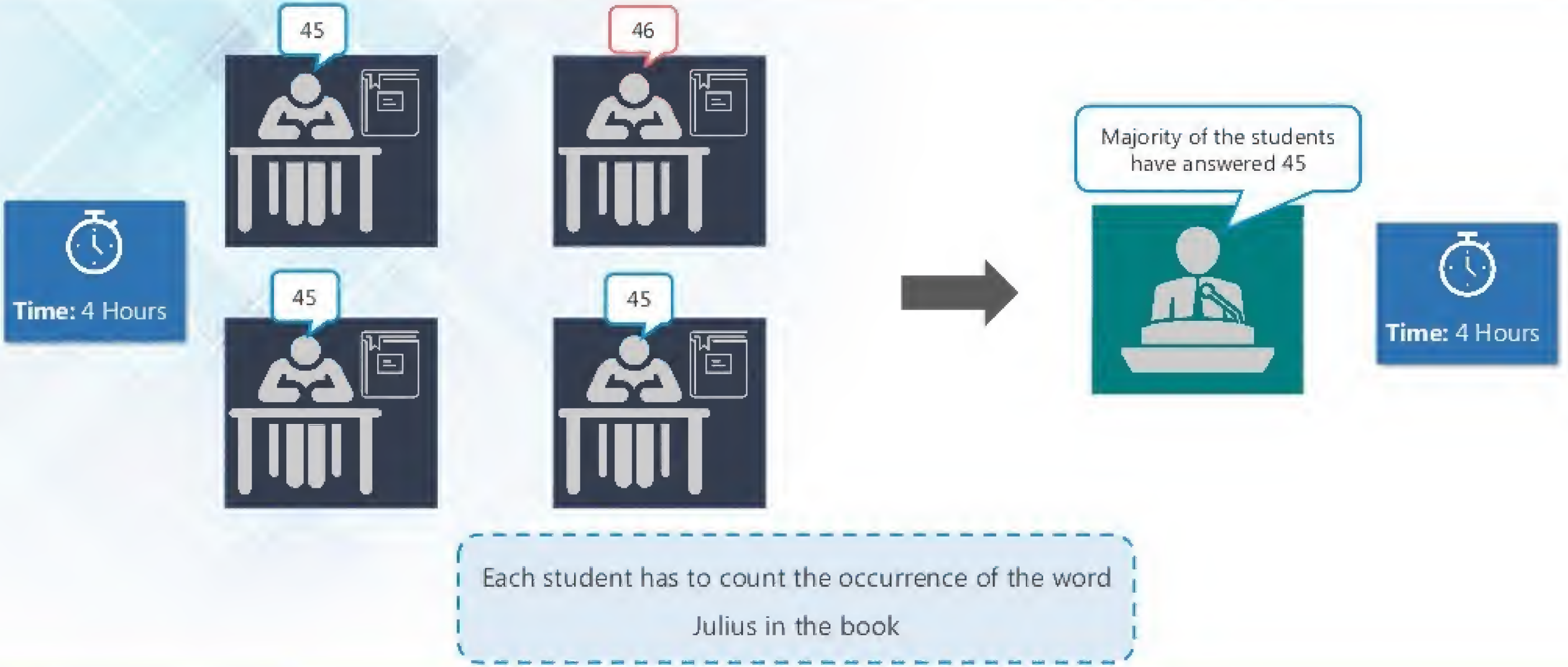
MapReduce



Processing:
Allows parallel & distributed processing

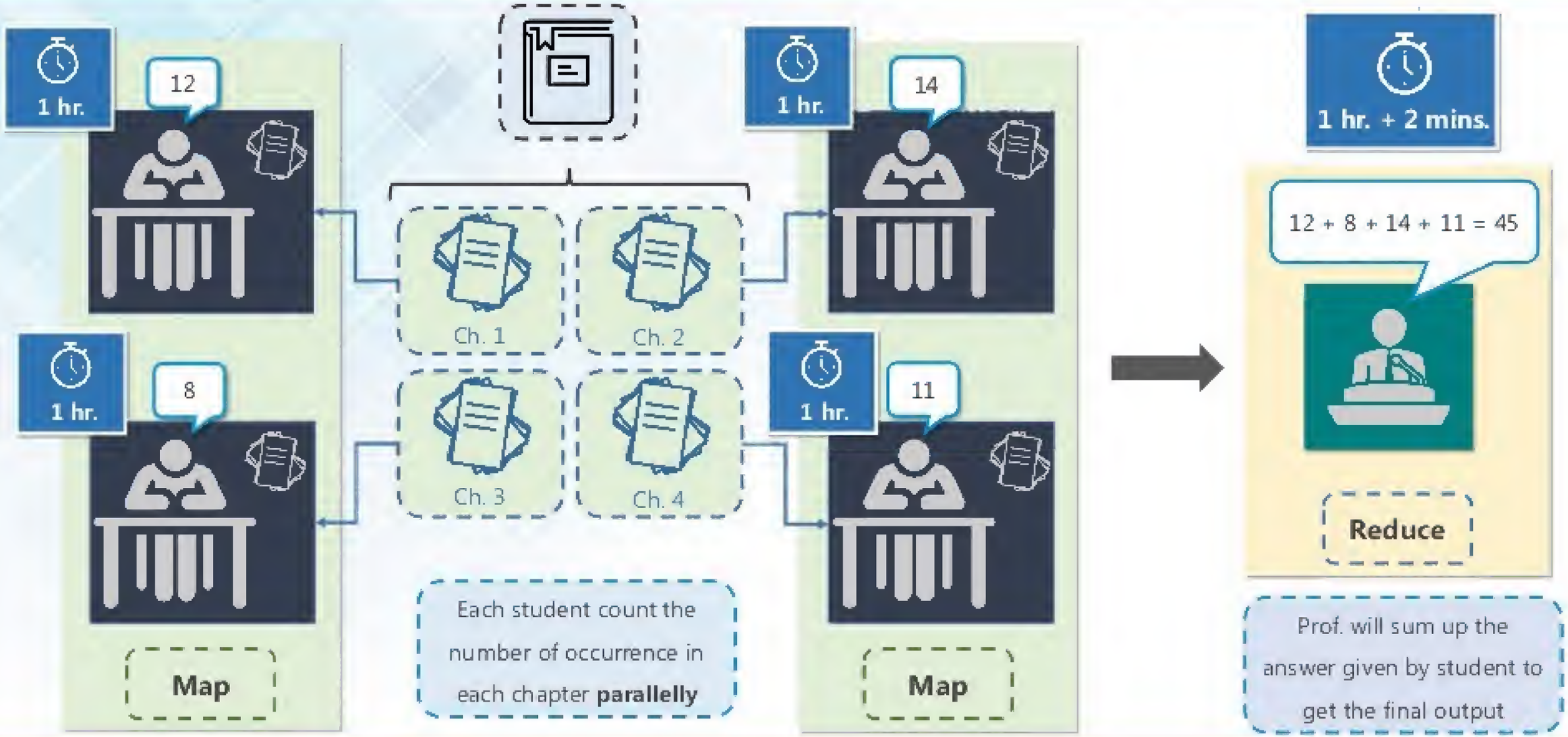
Let us understand MapReduce with a story

Story of MapReduce



Story of MapReduce

edureka!

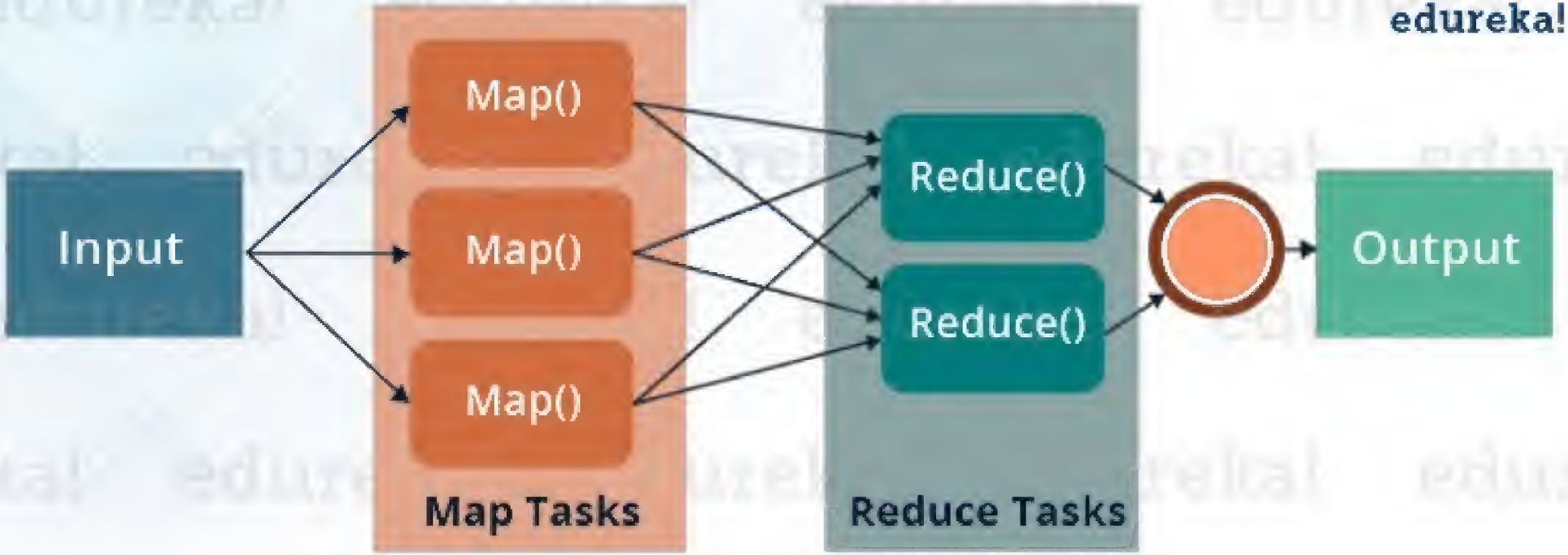


What is MapReduce?

What is MapReduce?

edureka!

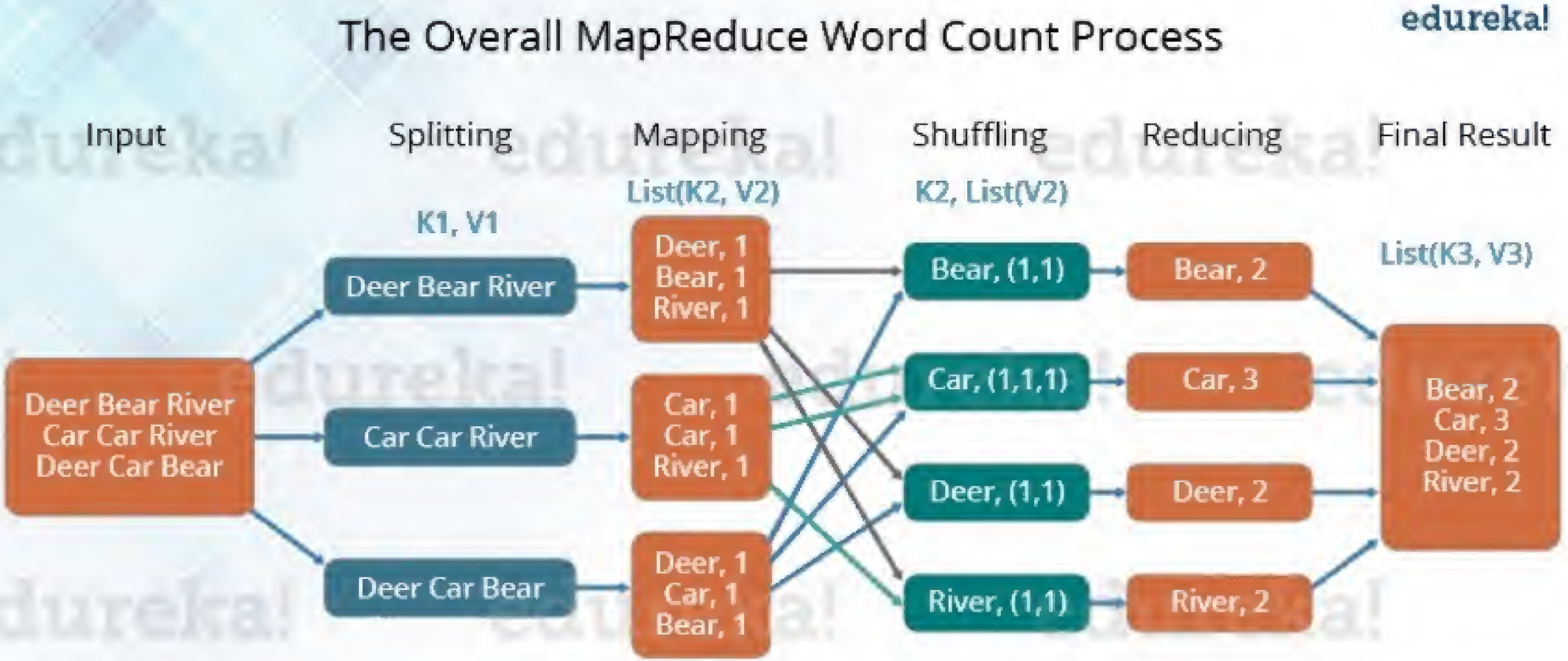
MapReduce is a programming framework that allows us to perform distributed and parallel processing on large data sets in a distributed environment



MapReduce
Word Count Program

MapReduce Word Count Program

edureka!



MapReduce Word Count Program

edureka!

Three Major Parts of MapReduce Program:

1

Mapper Code:

You write the mapper logic over here i.e. how map task will process the data to produce the key-value pair to be aggregated

2

Reducer Code:

You write reducer logic here which combines the intermediate key-value pair generated by Mapper to give the final aggregated output

3

Driver Code

You specify all the job configurations over here like job name, Input path, output path, etc.

Mapper Code

Byte Offset Type

Mapper Value Input Type

Mapper Key Output Type

Mapper Value Output Type

```
public static class Map extends Mapper<LongWritable,Text,Text,IntWritable> {

public void map(LongWritable key, Text value, Context context) throws IOException,InterruptedException {

String line = value.toString();
StringTokenizer tokenizer = new StringTokenizer(line);
while (tokenizer.hasMoreTokens()) {
value.set(tokenizer.nextToken());
context.write(value, new IntWritable(1));
}
```

Mapper Input:

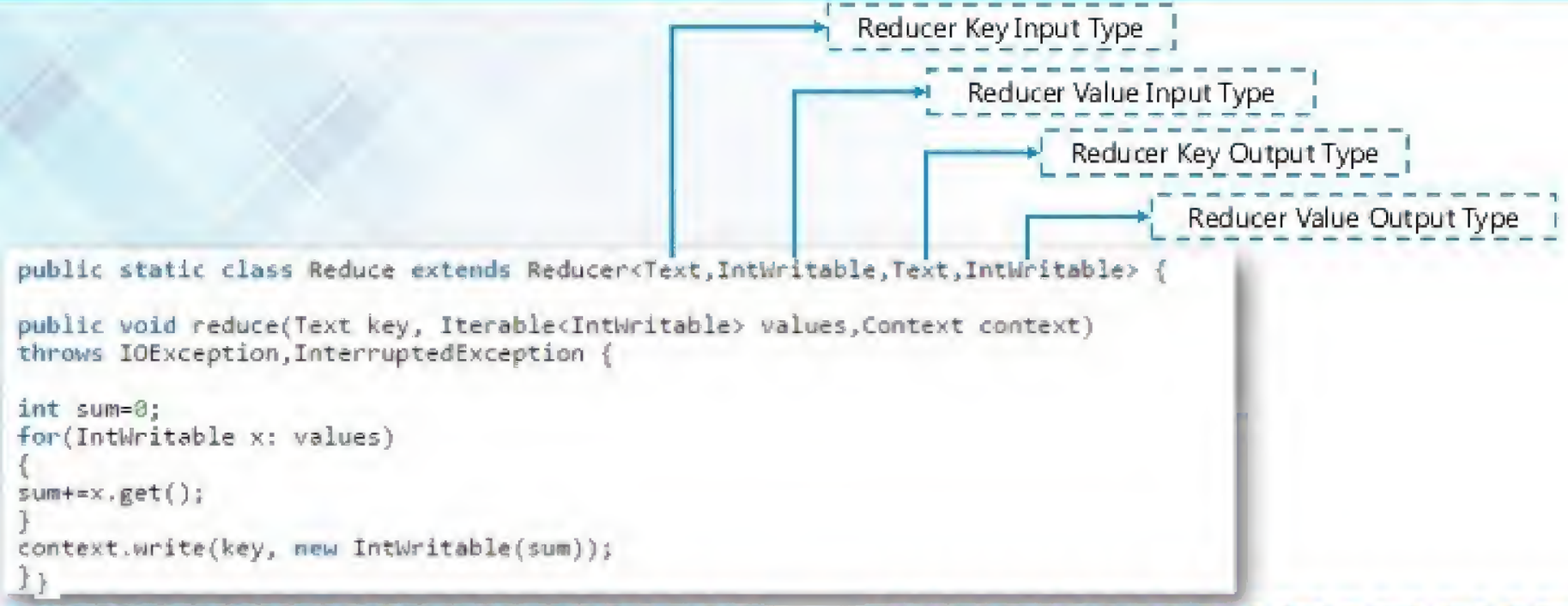
- The key is nothing but the offset of each line in the text file: *LongWritable*
- The value is each individual: *Text*

Mapper Output:

- The key is the tokenized words: Text
- We have the hardcoded value in our case which is 1: IntWritable
- Example – Dear 1, Bear 1, etc.

Reducer Code

edureka!



Reducer Input:

- Keys are unique words which have been generated after the sorting and shuffling phase: Text
- The value is a list of integers corresponding to each key: IntWritable
- Example: Bear, [1, 1], etc.

Reducer Output:

- The key is all the unique words present in the input text file: Text
- The value is the number of occurrences of each of the unique words: IntWritable
- Example: Bear, 2; Car, 3, etc. .

Driver Code

edureka!

In the driver class, we set the configuration of our MapReduce job to run in Hadoop

```
Configuration conf= new Configuration();
Job job = new Job(conf,"My Word Count Program");
job.setJarByClass(WordCount.class);
job.setMapperClass(Map.class);
job.setReducerClass(Reduce.class);
job.setOutputKeyClass(Text.class);

job.setOutputValueClass(IntWritable.class);
job.setInputFormatClass(TextInputFormat.class);
job.setOutputFormatClass(TextOutputFormat.class);
Path outputPath = new Path(args[1]);

//Configuring the input/output path from the filesystem into the job
FileInputFormat.addInputPath(job, new Path(args[0]));
FileOutputFormat.setOutputPath(job, new Path(args[1]));
```

- Specify the name of the job , the data type of input/output of the mapper and reducer
- Specify the names of the mapper and reducer classes.
- Path of the input and output folder
- The method setInputFormatClass () is used for specifying the unit of work for mapper
- Main() method is the entry point for the driver

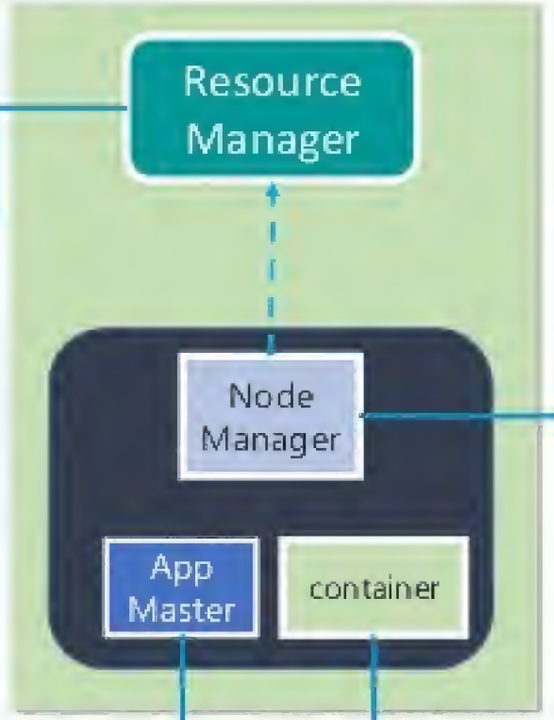
YARN Components

YARN Components

edureka!

ResourceManager:

- Master daemon that manages all other daemons & accepts job submission
- Allocates first container for the AppMaster



NodeManager:

- Responsible for containers, monitoring their resource usage i.e. (cpu, memory, disk, network) & reports the same to RM

AppMaster:

- One per application
- Coordinates and manages MR Jobs
- Negotiates resources from RM

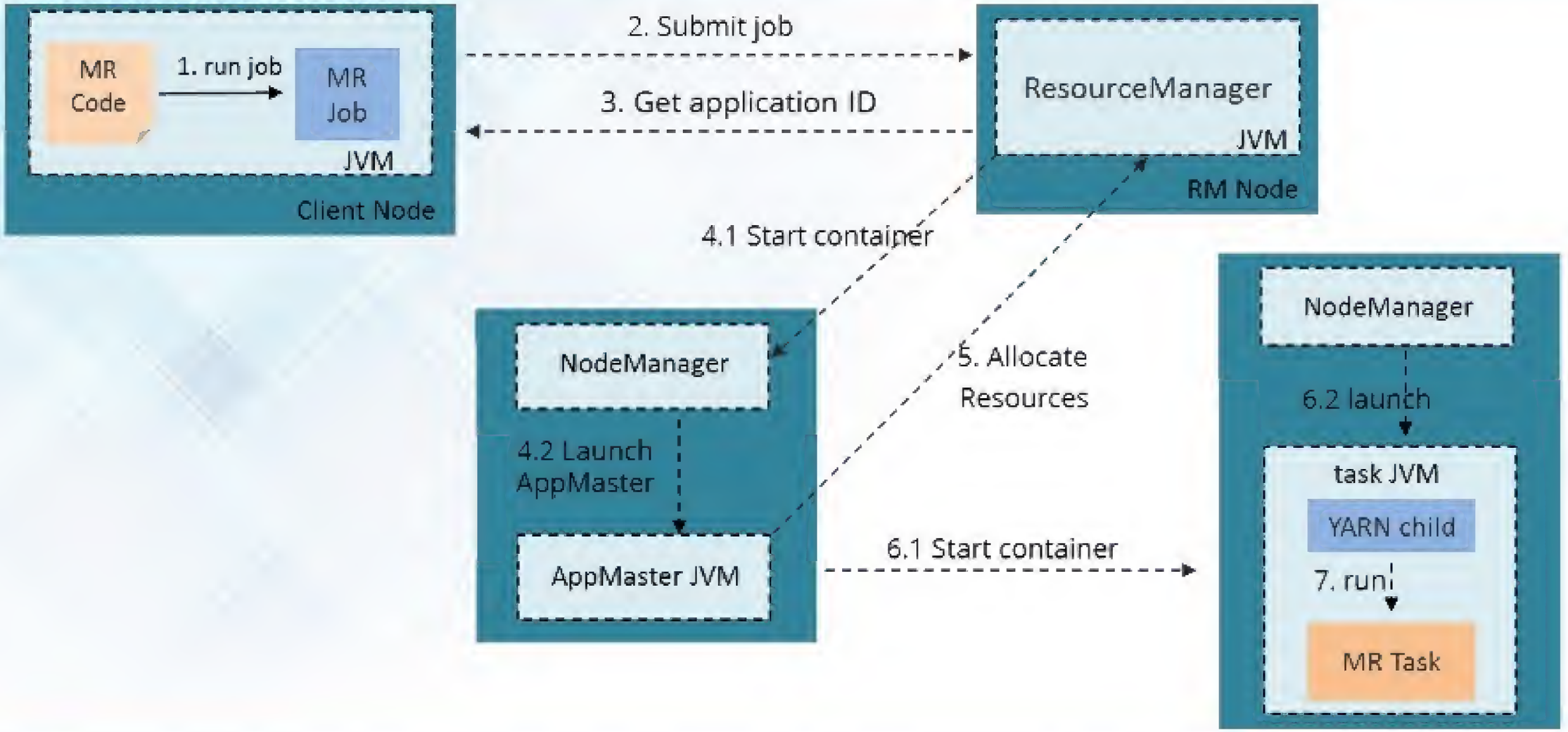
Container:

- Allocates certain amount of resources (memory, CPU etc.) on a slave node (NM)

MapReduce Job Workflow

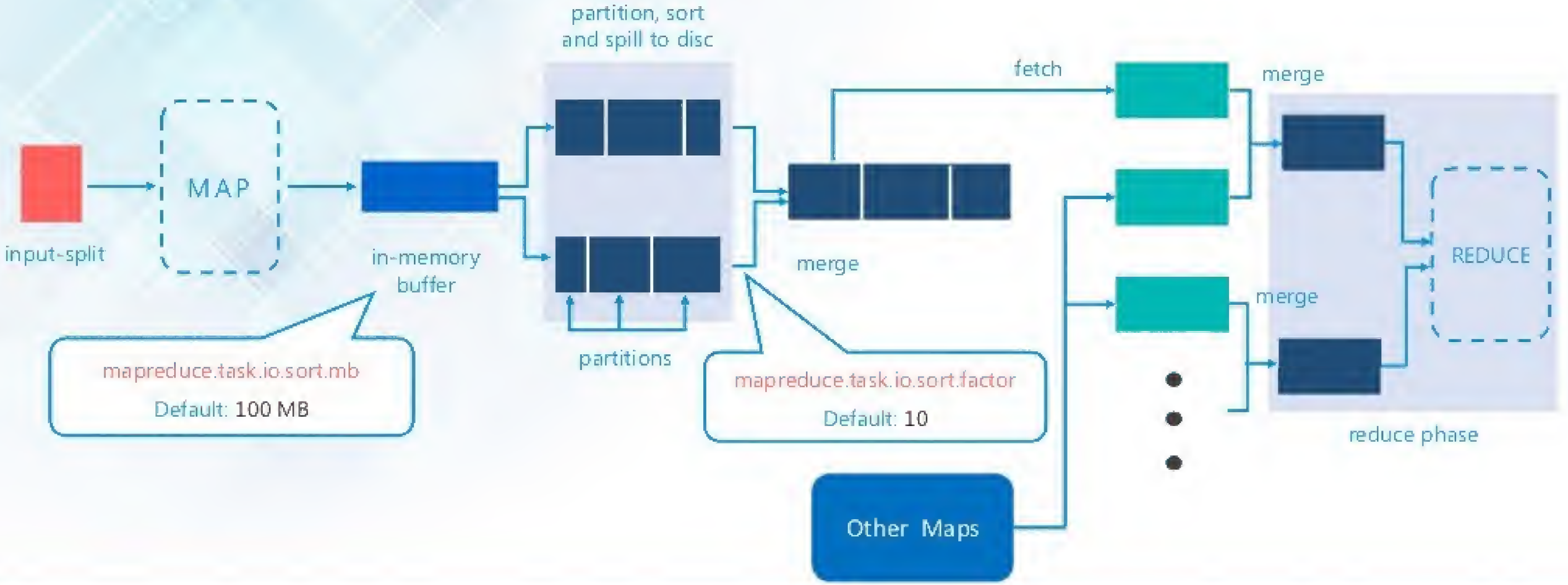
MapReduce Job Workflow

edureka!



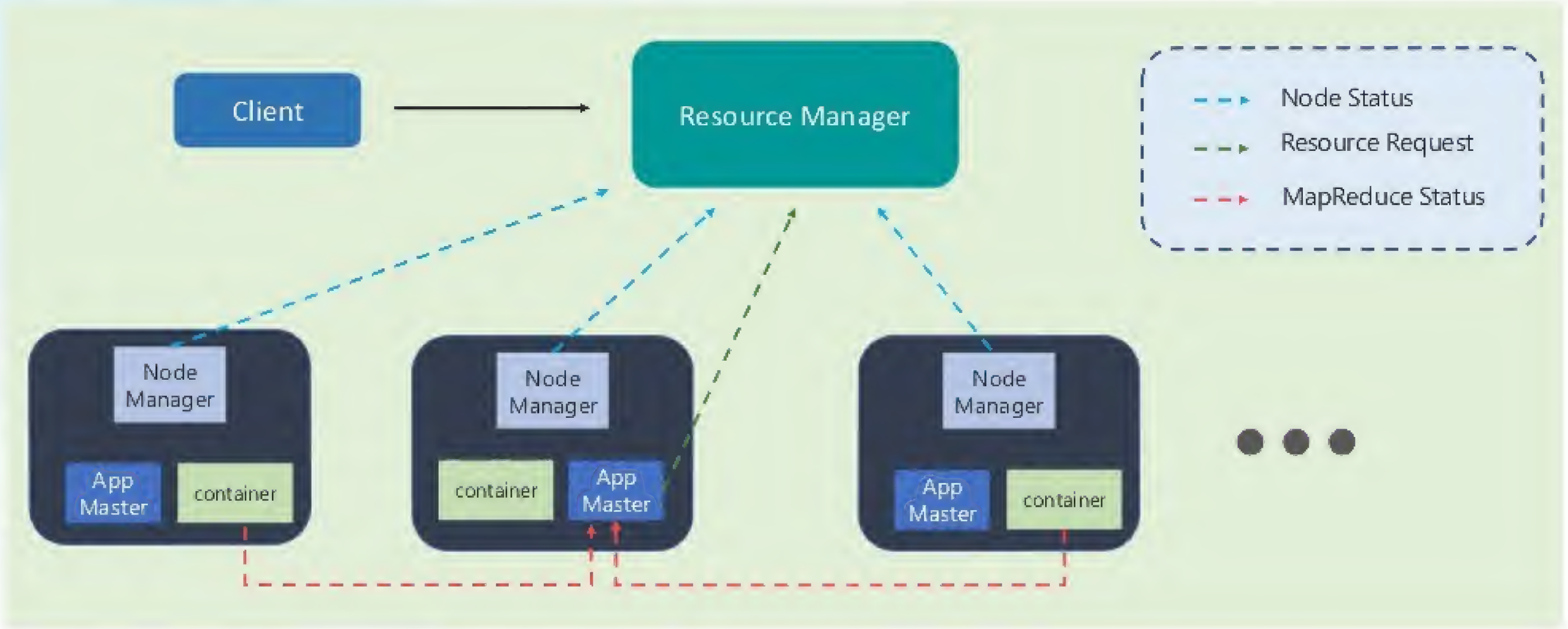
MAPREDUCE JOB WORKFLOW

edureka!



YARN Architecture

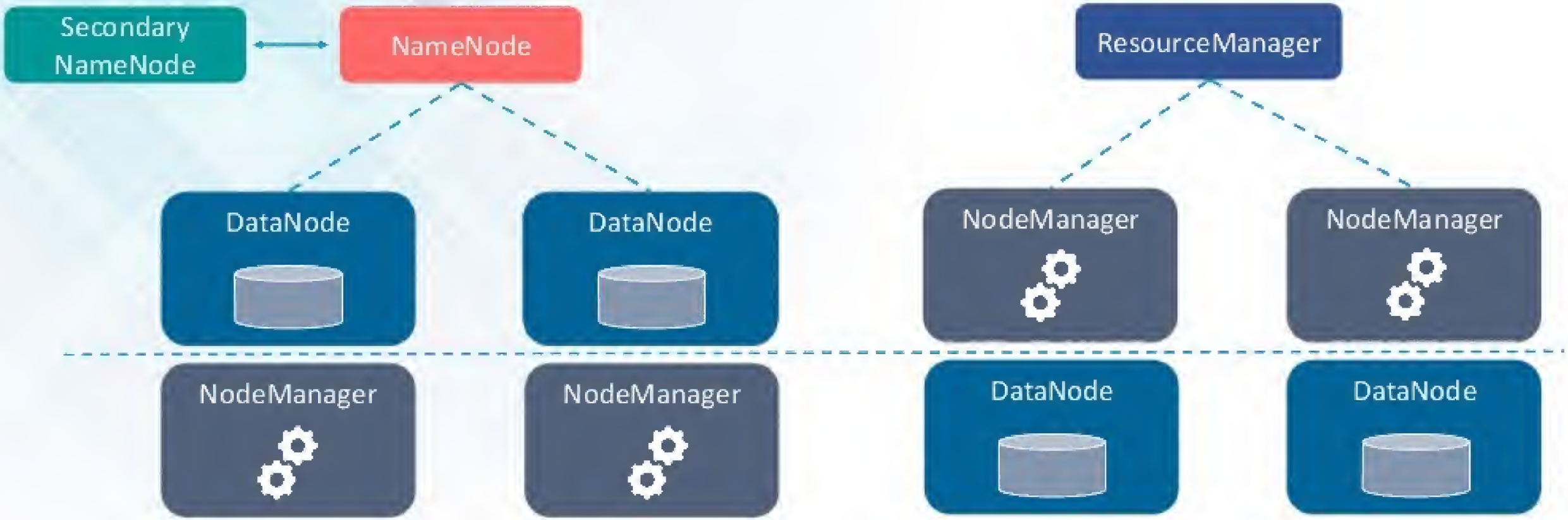
YARN Architecture



Hadoop Architecture: HDFS & YARN

Hadoop Architecture: HDFS & YARN

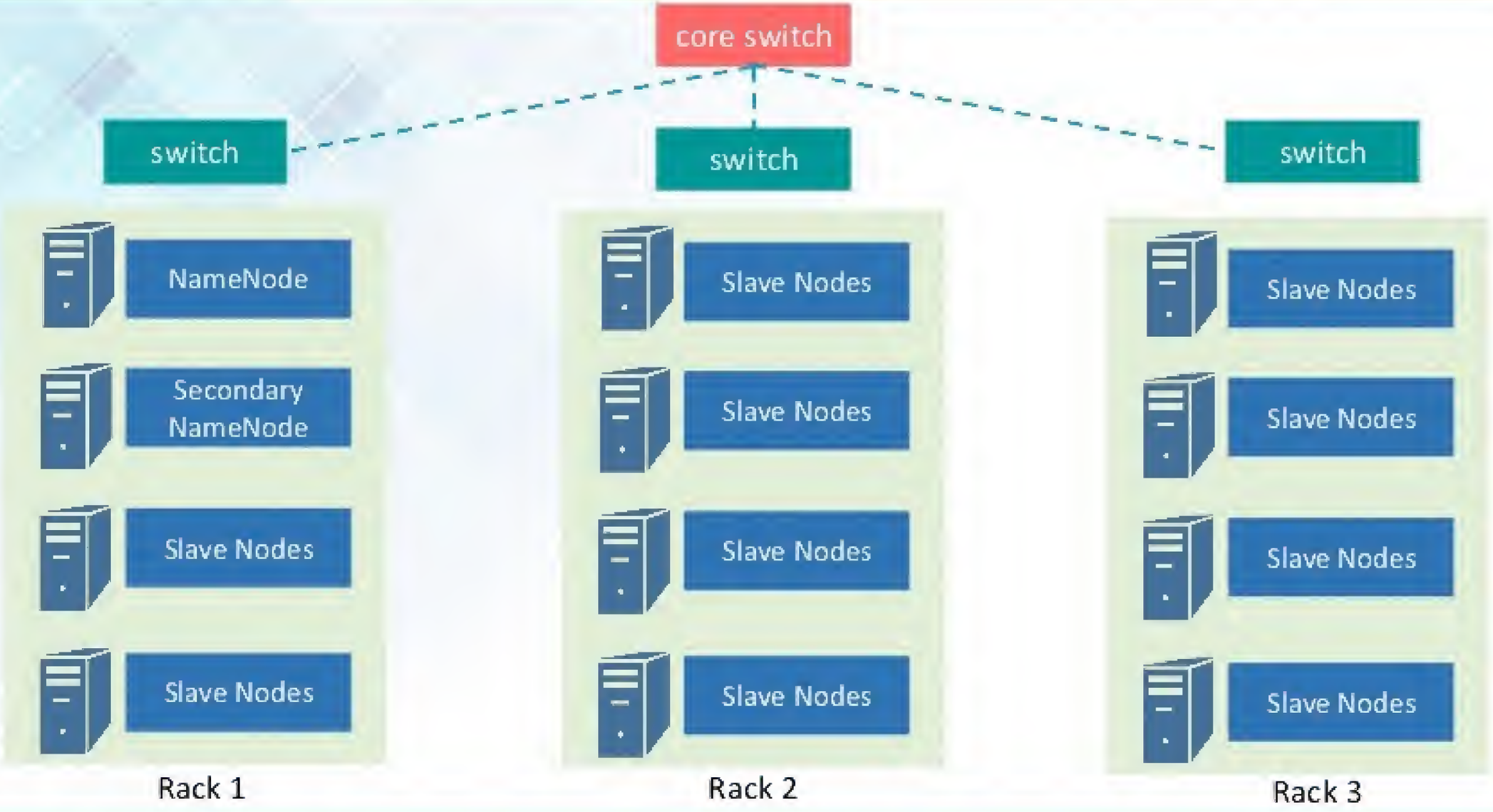
edureka!



Hadoop Cluster

Hadoop Cluster

edureka!



Hadoop Cluster Modes

Hadoop Cluster Modes

edureka!

Standalone (or Local) Mode

- No daemons, everything runs in a single JVM
- Suitable for running MapReduce programs during development
- Has no DFS or Distributed File System

Pseudo Distributed Mode

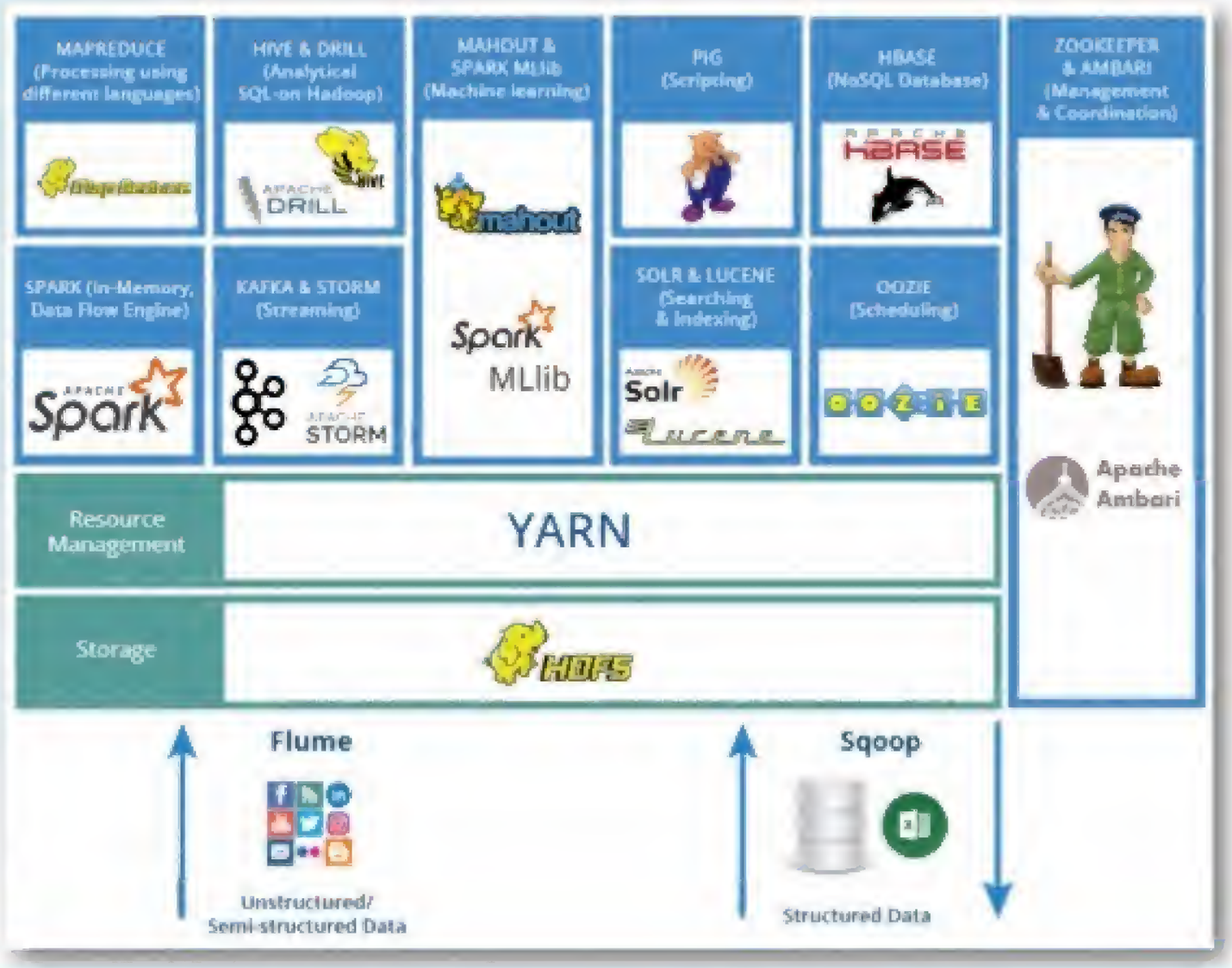
- All Hadoop daemons run on the local machine

Multi-Node Cluster Mode

- Hadoop daemons run on a cluster of machines

Hadoop Ecosystem

Hadoop Ecosystem



Hadoop Use Case: Analyzing Olympic Dataset

Hadoop Use Case: Analyzing Olympic Dataset

edureka!

Problem statement:

- Find the list of top 10 countries won the highest medals
- Find the total number of gold medals won by each country
- Which countries have won the most number of medals in swimming?

Dataset Description

edureka!

The data set consists of the following fields:

- **Athlete**: This field consists of the athlete name
- **Age**: This field consists of athlete ages
- **Country**: This fields consists of the country names which participated in Olympics
- **Year**: This field consists of the year
- **Closing Date**: This field consists of the closing date of ceremony
- **Sport**: Consists of the sports name
- **Gold Medals**: No. of Gold medals
- **Silver Medals**: No.of Silver medals
- **Bronze Medals**: No.of Bronze medals
- **Total Medals**: Consists of total no of medals

Dataset Description

edureka!

Athlete	Age	Country	Year	Closing Ceremony Date	Sport	Gold Medals	Silver Medals	Bronze Medals	Total Medals
Michael Phelps	23	United States	2008	08-24-08	Swimming	8	0	0	8
Michael Phelps	19	United States	2004	08-29-04	Swimming	6	0	2	8
Michael Phelps	27	United States	2012	08-12-12	Swimming	4	2	0	6
Natalie Coughlin	25	United States	2008	08-24-08	Swimming	1	2	3	6
Aleksey Nemov	24	Russia	2000	10-01-00	Gymnastics	2	1	3	6
Alicia Coutts	24	Australia	2012	08-12-12	Swimming	1	3	1	5
Missy Franklin	17	United States	2012	08-12-12	Swimming	4	0	1	5
Ryan Lochte	27	United States	2012	08-12-12	Swimming	2	2	1	5
Allison Schmitt	22	United States	2012	08-12-12	Swimming	3	1	1	5
Natalie Coughlin	21	United States	2004	08-29-04	Swimming	2	2	1	5
Ian Thorpe	17	Australia	2000	10-01-00	Swimming	3	2	0	5
Dara Torres	33	United States	2000	10-01-00	Swimming	2	0	3	5
Cindy Klassen	26	Canada	2006	02-26-06	Speed Skating	1	2	2	5
Nastia Liukin	18	United States	2008	08-24-08	Gymnastics	1	3	1	5
Marit Bjørgen	29	Norway	2010	02-28-10	Cross Country Skiing	3	1	1	5
Sun Yang	20	China	2012	08-12-12	Swimming	2	1	1	4
Kirsty Coventry	24	Zimbabwe	2008	08-24-08	Swimming	1	3	0	4
Libby Lenton-Trickett	23	Australia	2008	08-24-08	Swimming	2	1	1	4
Ryan Lochte	24	United States	2008	08-24-08	Swimming	2	0	2	4
Inge de Bruijn	30	Netherlands	2004	08-29-04	Swimming	1	1	2	4
Petria Thomas	28	Australia	2004	08-29-04	Swimming	3	1	0	4
Ian Thorpe	21	Australia	2004	08-29-04	Swimming	2	1	1	4
Inge de Bruijn	27	Netherlands	2000	10-01-00	Swimming	3	1	0	4
Gary Hall Jr.	25	United States	2000	10-01-00	Swimming	2	1	1	4
Michael Klim	23	Australia	2000	10-01-00	Swimming	2	2	0	4
Susie O'Neill	27	Australia	2000	10-01-00	Swimming	1	3	0	4

Demo


edureka!
Thank You
For more information please visit our website
www.edureka.co